НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Інститут телекомунікаційних систем

Кафедра Інформаційно-телекомунікаційних мереж

До захисту допущено:

Завідувач кафедри

_____ Лариса ГЛОБА

«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційно-комунікаційні технології»

спеціальності 172 «Телекомунікації та радіотехніка»

на тему: «Метод збору та обробки інформації в мережі індустріального Інтернету речей»

Виконав:
студент IV курсу, групи ТІ-62
Попенко Демид Віталійович _____

Керівник:
Асистент кафедри ІТМ ІТС
Курдеча Василь Васильович _____

Рецензент:
Старший викладач кафедри ТК ІТС, к.т.н.
Авдєєнко Гліб Леонідович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут телекомунікаційних систем
Кафедра Інформаційно-телекомунікаційних мереж

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 172 «Телекомунікації та радіотехніка»

Освітньо-професійна програма «Інформаційно-комунікаційні технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Лариса ГЛОБА
«___»_____2020 р.

ЗАВДАННЯ

на дипломну роботу студенту

Попенко Демиду Віталійовичу

1. Тема роботи «Метод збору та обробки інформації в мережі індустріального Інтернету речей», керівник роботи асистент кафедри інформаційно-телекомунікаційних мереж ІТС Курдеча Василь Васильович, затверджені наказом по університету від «30» березня 2020 р. № 924-с

2. Термін подання студентом роботи 8 червня 2020 р.

3. Вихідні дані до роботи: теоретичні матеріали про методи збору та обробки інформації в мережі IIoT. План розробки дипломної роботи

4. Зміст роботи

1) Проаналізувати особливості IIoT.

2) Проаналізувати проблеми IIoT та здійснити огляд існуючих рішень.

3) Модифікувати метод збору та обробки інформації в мережі IIoT.

4) Провести натурне моделювання запропонованого рішення

5) Провести оцінку модифікованого методу збору та обробки інформації

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

1) Актуальність, мета роботи, основні задачі;

2) Наукова новизна, практична цінність;

3) Порівняння IoT та IIoT, аналіз проблем IIoT;

4) Порівняння існуючих рішень;

5) Модифікований метод збору та обробки інформації в мережі IIoT;

6) Порівняння методів серіалізації;

7) Натурна модель, натурне моделювання запропонованого рішення;

8) Процес моделювання, оцінка модифікованого методу;

9) Загальні висновки по роботі.

6. Дата видачі завдання 3 жовтня 2019 року

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Вступ	14.10.2019 – 21.10.2019	Виконано
2	Огляд основних відмінностей IIoT та IoT. Аналіз особливостей побудови IIoT.	22.10.2019 – 18.11.2019	Виконано
3	Аналіз проблем IIoT. Огляд існуючих рішень	19.11.2019 – 23.12.2019	Виконано
4	Запропонування модифікованого методу	24.12.2019 – 20.01.2020	Виконано
5	Проведення моделювання запропонованого рішення за визначеними критеріями	21.01.2020 – 24.02.2020	Виконано
6	Провести оцінку запропонованого рішення	25.02.2020 – 08.03.2020	Виконано
7	Підготовка та участь у міжнародних конференціях за темою бакалаврської роботи	09.03.2020 – 17.04.2020	Виконано
8	Сформувати математичний опис отриманих результатів за кожним критерієм	18.04.2020 – 17.05.2020	Виконано
9	Оформлення роботи	18.05.2020 – 08.06.2020	Виконано

Студент Демид ПОПЕНКО

Керівник Василь КУРДЕЧА

РЕФЕРАТ

Робота містить 72 сторінки, 35 рисунки, 13 таблиць. Було використано 27 джерел.

Актуальність роботи полягає в тому, що при впроваджені розумного виробництва отриману з пристроїв інформацію необхідно обробити, що вимагає значних часових витрат та пам’яті для зберігання накопиченої інформації. Через це постає задача мінімізації часу обробки і розміру файлу для передачі по мережі.

Метою роботи є підвищення ефективності збору та обробки інформації в мережі індустріального Інтернету речей за рахунок використання модифікованого методу.

В процесі виконання роботи досягли збільшення швидкості обробки інформації, одержуваної від устаткування IIoT, за рахунок використання механізму серіалізації, а також зменшення розміру файлу з інформацією, яка передається по мережі.

Ключові слова: IIoT, збір інформації, обробка інформації, серіалізації, Protobuf.

ABSTRACT

The work contains 72 pages, 35 figures, 13 tables. 27 sources were used.

The relevance of the work lies in the fact that when implementing intelligent production, information received from devices should be processed, which requires significant time and memory to store the accumulated information. Because of this, the problem arises of minimizing the processing time and file size for transmission over the network.

The aim of the work is to increase the efficiency of collecting and processing information in the industrial Internet of Things through the use of a modified method.

In the course of the work, the speed of processing the information received from the IIoT equipment was increased due to the use of the serialization mechanism, as well as the reduction of the file size with the information transmitted over the network.

Key words: IIoT, information collection, information processing, serialization, Protobuf.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

IoT	Internet of Things
IIoT	Industrial Internet of Things
CPS	Cyber Physical System
IT	Information Technology
OT	Operational Technology
API	Application Programming Interface
DDS	Data Distribution Service
MQTT	Message Queuing Telemetry Transport
CoAP	Constrained Application Protocol
BLE	Bluetooth Low Energy
XML	eXtensible Markup Language
JSON	JavaScript Object Notation
Protobuf	Protocol Buffers
QoS	Quality of Service

ВСТУП

Актуальність. Отриману з пристроїв на підприємстві інформацію необхідно обробити, що вимагає значних часових ресурсів та пам’яті для зберігання накопиченої інформації. Також різні види пристроїв мають різні типи даних, які в сукупності являють собою масивні неоднорідні набори інформації з багатьох джерел. Все це призводить до великих витрат на збір та обробку інформації. Через це постає задача модифікації методу збору і обробки інформації, який забезпечить мінімальний час обробки і розмір файлу для передачі по мережі.

Мета й завдання дослідження

Метою роботи є підвищення ефективності збору та обробки інформації в мережі індустріального Інтернету речей за рахунок використання модифікованого методу.

В роботі для досягнення поставленої мети були вирішені такі **основні задачі:**

1) Проаналізувати особливості IIoT;
2) Проаналізувати особливості IIoT;
3) Модифікувати метод збору та обробки інформації в мережі IIoT;
4) Провести натурне моделювання запропонованого рішення;
5) Провести оцінку модифікованого методу збору та обробки інформації.

Об’єкт дослідження - процес збору та обробки інформації в мережі індустріального Інтернету речей.

Предмет дослідження - метод збору та обробки інформації.

Теоретичний результат роботи:

1) Аналіз особливостей та проблем IIoT;
2) Модифікований метод збору і обробки інформації за рахунок використання механізму серіалізації.

Практичний результат роботи:

1) На основі модифікованого методу збору і обробки інформації реалізовано програмне забезпечення для обробки вхідної інформації.

РОЗДІЛ 1
АНАЛІЗ ОСОБЛИВОСТЕЙ IIOT

1.1. Порівняння IIoT та IoT

IoT та IIoT тісно пов'язані між собою поняття, але їх не можна взаємозамінно використовувати. Щодо IoT, існує декілька визначень, кожне з яких намагається зафіксувати одну зі своїх основних характеристик. Його часто вважають своєрідною мережею для машин, підкреслюючи мету обміну даними. Однак сфери застосування настільки різноманітні, що деякі вимоги, особливо ті, що стосуються аспектів комунікації, можуть бути дуже різними, залежно від намічених цілей та кінцевих споживачів, базових бізнес-моделей та прийнятих технологічних рішень. Те, що зазвичай називається IoT, краще назвати користувацьким IoT, а не промисловим IoT [1, 2].

Користувальницький IoT орієнтований на людину; «речі» - це розумні електронні пристрої, пов'язані між собою з метою покращення обізнаності людей про навколишнє середовище, заощадження часу та грошей. Загалом, користувальницькі IoT-комунікації можуть бути класифіковані як людина-машина та у формі взаємодії клієнт-сервер.

З іншого боку, в індустріальному світі ми допомагаємо появі цифрового та інтелектуального виробництва, метою якого є інтеграція операційних технологій (OT) у сфери інформаційних технологій (IT).

IIoT - це з'єднання всіх промислових активів, включаючи машини та системи управління, з інформаційними системами та бізнес-процесами. Як наслідок, велика кількість зібраних даних може забезпечити аналітичні рішення і привести до оптимальних виробничих операцій. З іншого боку, інтелектуальне виробництво, очевидно, фокусується на стадії виробництва життєвого циклу (інтелектуальних) продуктів з метою швидкого та динамічного реагування на зміни попиту. Таким чином, IIoT впливає на весь виробничий ланцюг формування вартості і є вимогою для розумного виробництва.

Комунікація в IIoT орієнтована на машини і може охоплювати широкий спектр різних секторів ринка і видів діяльності. Сценарії IIoT включають традиційні додатки моніторингу (наприклад, моніторинг процесів на виробничих підприємствах) і інноваційні підходи до самоорганізованих систем (наприклад, автономні промислові підприємства, які практично не потребують втручання людини) [3].

У той час як вимоги IoT і IIoT для зв'язку аналогічні, наприклад, підтримка екосистеми Інтернету з використанням недорогих пристроїв з обмеженими ресурсами і масштабованість мережі, багато вимог до зв'язку специфічні для кожного домена і можуть бути дуже різними, наприклад, якість обслуговування (з точки зору затримки, пропускної спроможності і т. д.), доступність і надійність, а також безпека і конфіденційність. IoT більше уваги приділяє розробці нових стандартів зв'язку, які можуть підключати нові пристрої до екосистеми Інтернету гнучким і зручним для користувача способом. IIoT навпаки робить упор на можливу інтеграцію і взаємозв'язок колись ізольованих заводів або навіть механізмів, пропонуючи тим самим більш ефективне виробництво і нові послуги [3]. З цієї причини, в порівнянні з IoT, IIoT можна вважати скоріше еволюцією, ніж революцією.

Що стосується підключення і критичного режиму, IoT є більш гнучким, дозволяючи створювати структури ad hoc і мережі мобільного зв'язку, а також пред'являти менш суворі вимоги до синхронізації і надійності. З іншого боку, IIoT зазвичай використовує фіксовані і засновані на інфраструктурі мережеві рішення, які розроблені для відповідності потребам в комунікації і співіснуванні. У IIoT зв'язок здійснюється в формі міжмашинних з'єднань, які повинні задовольняти строгим вимогам з точки зору своєчасності та надійності.

Згенеровані дані в IoT сильно залежать від додатків, в той час як IIoT орієнтований на аналітику, наприклад, для профілактичного обслуговування і поліпшення логістики.

Концепція Індустрії 4.0 (де 4.0 представляє четверту промислову революцію) виникає, коли парадигма IoT об'єднується з ідеєю кібер-фізичних систем (CPS) [4].

Сформована в Німеччині, концепція Індустрії 4.0 отримала глобальну популярність, і в даний час вона універсально прийнята для вирішення проблеми використання інтернет-технологій для підвищення ефективності виробництва за допомогою інтелектуальних послуг на інтелектуальних фабриках. CPS розширюють фізичні об'єкти реального світу, повністю об'єднуючи їх і надаючи їм цифровий опис. Інформація, що зберігається в моделях і об'єктах даних, які можуть бути оновлені в режимі реального часу, являє собою другу ідентичність самого об'єкта і являє собою свого роду «цифрового близнюка». Завдяки динамічній природі цих цифрових близнюків, інноваційні послуги, які були неможливі в минулому, можуть бути реалізовані протягом усього життєвого циклу продукту, від створення до утилізації виробленої продукції.

На рисунку 1.1 наведено базові напрямки реалізації Індустрії 4.0

Рис. 1.1 Базові напрямки реалізації Індустрії 4.0

Індустрія 4.0 включає в себе реалізації наступних напрямків:

1) Індустріальний інтернет речей (датчики й сенсори на виробництві об'єднані в єдину мережу з ієрархічною структурою і підпорядковані єдиній системі управління виробництвом);

2) Великі дані (big data) і бізнес аналітика (аналітика, яка спирається на великі);
3) «Хмарні технології», необхідність в обробці великих баз даних вимагає постійного вдосконалення «хмарних сервісів»;
4) Інформаційна безпека (захищений доступ та надійний зв'язок до мереж управління);
5) Адитивне виробництво (наприклад застосування 3D-друку);
6) Горизонтальна і вертикальна інтеграція систем (створення середовища тісної взаємодії як на різних рівнях у межах одного підприємства так і між підприємствами);
7) Цифрове моделювання виробничих процесів.

Таблиця 1.1

Порівняння IIoT та IoT

	Користувальницький IoT	Індустріальний IoT
Передумови	Революція	Еволюція
Сервісна модель	Людино-орієнтований	Машинно-орієнтований
Поточний статус	Нові пристрої та стандарти	Існуючі пристрої та стандарти
Зв’язок	Структури ad hoc (вузли можуть бути мобільними)	Структуровані (вузли фіксовані; централізоване управління мережею)
Критичність (умови протікання процесу)	Не суворі (виключаючи медичні додатки)	Критичне значення (час, надійність, безпека, конфіденційність)
Обсяг даних	Від середнього до високого	Від високого до дуже високого

Таким чином, IIoT є підмножиною IoT. Рисунок 1.2 показує перетин IoT, CPS, IIoT і Індустрії 4.0.

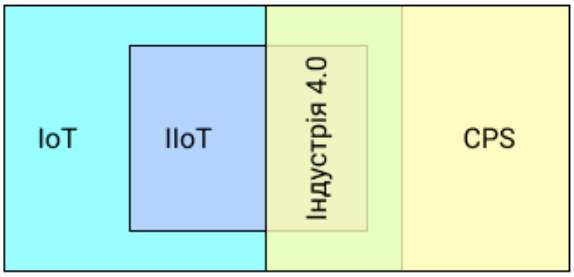

Рис. 1.2 Перетин IoT, CPS, IIoT і Індустрії 4.0

1.2. Аналіз особливостей побудови IIoT

Рівні архітектури IIoT визначаються шляхом аналізу різних варіантів використання IIoT, розроблених міжнародним інформаційним центром і визначення відповідних зацікавлених сторін систем IIoT [5].

Як показано на рисунку 1.3, чотири точки зору складають основу для детального аналізу окремих систем IIoT.

Рис. 1.3 Точки зору IIoT

Точка зору бізнесу

Точка зору бізнесу враховує інтереси зацікавлених сторін і їх бізнес-бачення, цінності та цілі при створенні системи IIoT в діловому і нормативному контексті. Крім того, в ньому визначається, яким чином система IIoT досягає поставлених цілей за допомогою її зіставлення з

фундаментальними можливостями системи. На рисунку 1.4 представлена ця точка зору, яка представляє особливий інтерес для керівників підприємств, менеджерів продуктів і системних інженерів.

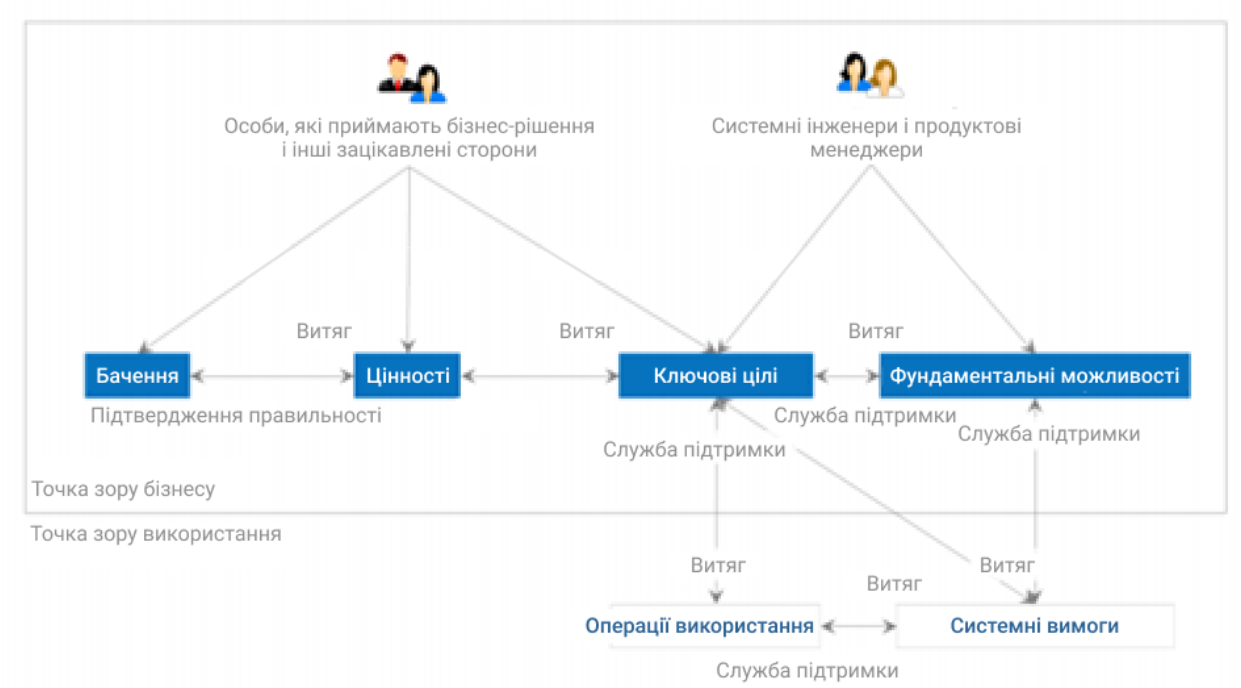

Рис. 1.4 Точка зору бізнесу

Точка зору використання

Точка зору використання стосується того, як система IIoT реалізує ключові можливості, визначені в рівні бізнесу. Як правило, вона представлена у вигляді послідовностей дій за участю людей, які забезпечують її призначену функціональність для досягнення, в кінцевому рахунку, її фундаментальних системних можливостей. На рисунку 1.5 представлена ця точка зору у якій, як правило, зацікавленими сторонами є системні інженери, менеджери продуктів та особи, які беруть участь у специфікації розглянутої системи IIoT.

Рис. 1.5 Точка зору використання

Функціональна точка зору

Функціональна точка зору фокусується на функціональних компонентах системи IIoT, їх структурі та взаємозв'язку, інтерфейсах та взаємодіях між ними, а також на взаємодії системи із зовнішніми елементами в навколишньому середовищі для підтримки використання та діяльності загальної системи. Ця точка зору представляє особливий інтерес для системних та компонентних архітекторів, розробників та інтеграторів.

Декомпозиція типової системи IIoT на функціональні домени виділяє важливі формуючі блоки, які мають широке застосування у багатьох промислових вертикалях. Це початкова точка для концептуалізації конкретної функціональної архітектури. Конкретні системні вимоги сильно впливатимуть на те, як функціональні домени будуть декомпоновані, які додаткові функції можуть бути додані чи прибрані, а також які функції можна комбінувати та додатково розкласти.

Типова система IIoT розкладається на п'ять функціональних доменів (див. рис. 1.6) [6]:

- Керуючий домен;
- Домен операцій;

- Інформаційний домен;
- Домен додатків;
- Домен бізнесу.

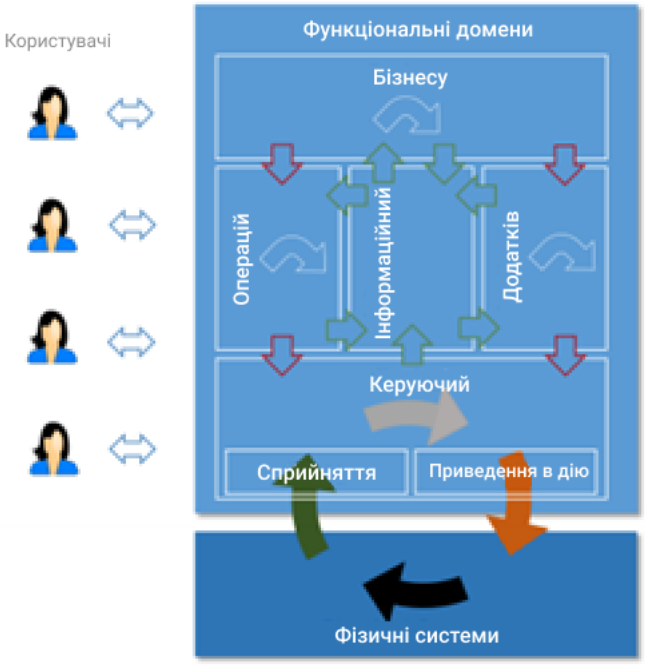

Рис. 1.6 Функціональні домени

Керуючий домен фокусується на функціях датчиків та механізмів (див. рис. 1.7). Взаємодія із зовнішніми фізичними об'єктами та навколишнім середовищем є головним аспектом цього домена, що також стосується екологічної безпеки, стійкості та захисту даних.

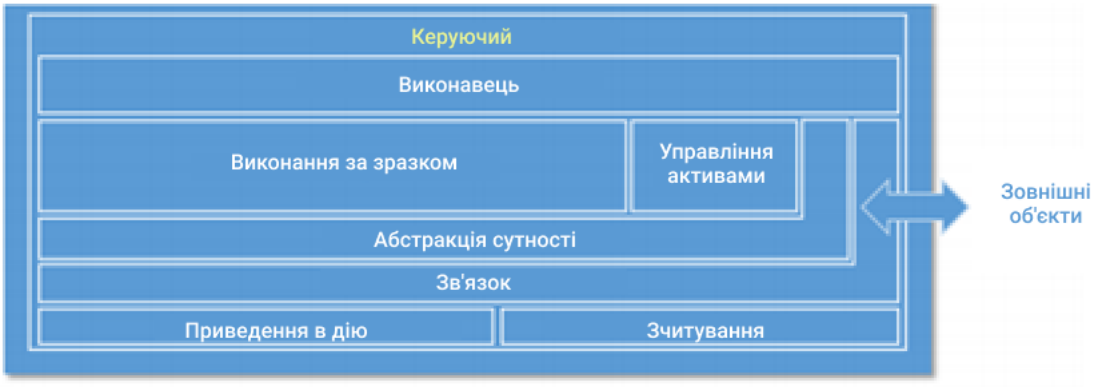

Рис. 1.7 Керуючий домен

Домен операцій. В архітектурі IIoT традиційні промислові засоби управління, які, як правило, зосереджені на одному місцевому фізичному заводі, переходять на більш високий рівень (див. рис. 1.8). Домен операцій включає функції щодо забезпечення, управління, моніторингу та оптимізації для багатьох установок, типів активів або клієнтів.

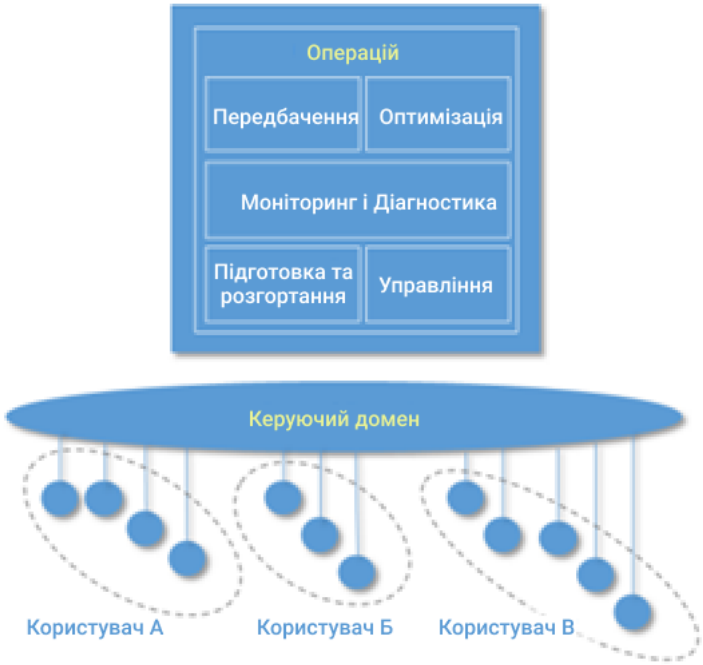

Рис. 1.8 Домен операцій

Інформаційний домен являє собою сукупність функцій збору даних з різних доменів, здебільшого з керуючого домену. Потім ці дані перетворюються, зберігаються та моделюються для отримання інформації про загальну систему; що, в свою чергу, допомагає нам отримувати інформацію, яка спирається на дані та динамічну оптимізацію. Наприклад, використовуючи витрати, попит та логістику, продуктивність автоматизованого виробництва може динамічно змінюватися.

Домен додатків включає функції для реалізації бізнес-функцій, таких як логіка додатків та правила, API, інформаційні панелі тощо.

Домен бізнесу. Функції інтегрують системи IIoT з традиційними або новими бізнес-додатками, такими як ERP, CRM, управління життєвим циклом продукту (PLM), система управління виробництвом (MES), управління людськими ресурсами (HRM), управління активами, управління життєвим циклом обслуговування, виставлення рахунків та системи оплати, планування роботи та розклад систем тощо (див. рис. 1.9).

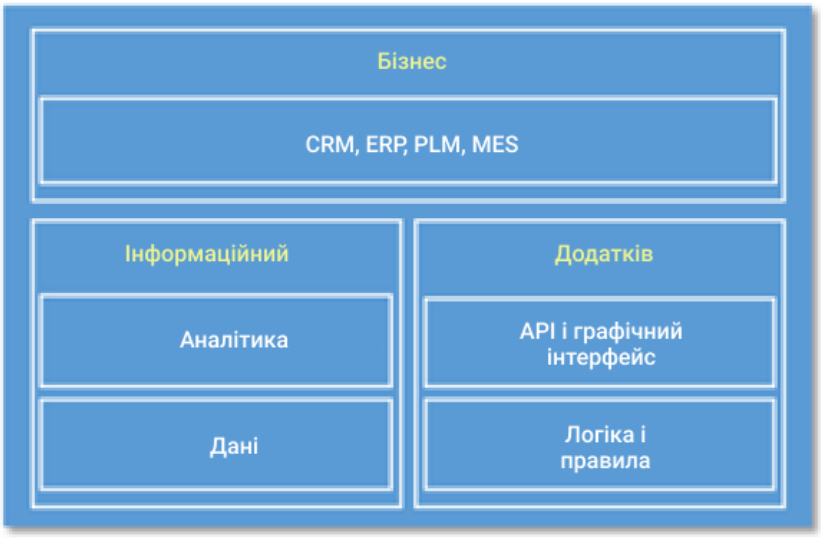

Рис. 1.9 Домен бізнесу

Точка зору реалізації

Точка зору реалізації стосується технологій, необхідних для впровадження функціональних компонентів (функціонального рівня), їхні схеми зв'язку та терміни експлуатації. Ці елементи координуються точкою зору використання та підтримують можливості системи бізнес рівня. Це представлення особливо цікаве для системних та компонентних архітекторів, розробників, інтеграторів та системних операторів.

Архітектурні патерни домену реалізації

Архітектурний патерн - це спрощене і абстрактне представлення реалізації системи IIoT, яке періодично зустрічається в багатьох системах IIoT.

Послідовні реалізації системи IIoT слідують певним усталеним архітектурним схемами, таким як:

- Трирівнева архітектура;
- Взаємодія і управління за допомогою шлюзу;
- Багаторівнева шина даних.

Висновки:

1) Розглянуто та проведено порівняння IIoT та IoT. Проведений аналіз особливостей побудови IIoT, детально розглянуті точки зору, які складають основу для аналізу систем IIoT, що дозволило виявити існуючі проблеми IIoT.

РОЗДІЛ 2
МЕТОДИ ЗБОРУ І ОБРОБКИ ІНФОРМАЦІЇ В МЕРЕЖІ ІІОТ

2.1. Аналіз проблем ІІоТ

Провідні виробничі компанії в різних географічних регіонах, включаючи США, Європу і Азіатсько-Тихоокеанський регіон, постійно розширюють свої виробничі операції і впроваджують інтелектуальні технології виробництва.

Впровадження ІІоТ дозволяє користувачам використовувати дані і аналітику для прогнозованого аналізу, скорочення часу простою обладнання, централізоване зберігання і віддалений моніторинг активів. Проте ІІоТ має власні проблеми, які виробники і підприємства повинні вирішити, щоб користуватися перевагами зв’язаного виробництва [7].

На рисунку 2.1 перераховані основні проблеми ІІоТ.

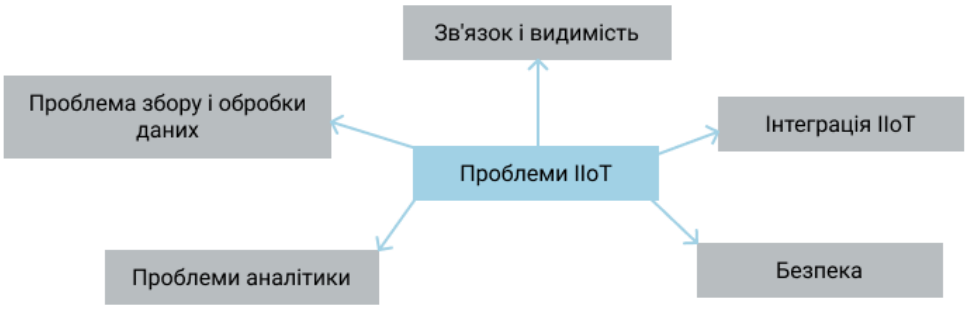

Рис. 2.1 Проблеми ІІоТ

Зв'язок і видимість

Критичні проблеми реалізації ІІоТ полягають у відсутності можливості підключення. Управління машинами ІІоТ є критично важливим питанням, і вкрай важливо стежити за машинами в режимі реального часу, а також стежити за тим, щоб ці машини працювали на оптимальному рівні для підвищення продуктивності. Підвищена видимість і краще розуміння стану машини також мають вирішальне значення для виявлення аномалій і усунення проблем до їх виникнення.

Датчики IIoT підключені до різних компонентів, і часто виникають проблеми з синхронізацією через перебої в мережі, відключення живлення і ручних або технічних помилок. Це призводить до відсторонення підключених пристроїв від мережі, що впливає на весь виробничий процес і може спричинити значні збитки.

Інтеграція IIoT

Ще одне складне завдання, яке виникає перед впровадженням IIoT, - це інтеграція інформаційних технологій (IT) і операційних технологій (OT). Дуже важливо забезпечити надійну інтеграцію цих двох компонентів без втрати даних та не допустити появи вразливості. Це робить впровадження технології IIoT фінансово недоцільним з точки зору логістики для багатьох діючих підприємств.

В основному пристрої IIoT зазвичай розробляються як незалежні рішення, і в кращих випадках вони впроваджуються у виробничий процес, щоб стати частиною системи. Однак інтеграція між інформаційними технологіями (IT) і експлуатаційними технологіями (OT) не забезпечує ефективного зв'язку та синхронізації. Отже, необхідно або замінити все своє обладнання, або покладатися на дефектний зв'язок.

Безпека

Проблеми безпеки для технологій IIoT є однією з найбільших проблем, оскільки це зачіпає як окремих осіб, так і організації, вразливі до фінансових і операційних збитків. Як промислові компанії, що використовують різні рішення IIoT, вони відкриті для проблем безпеки, відсутності видимості устаткування, уразливості злому, ризиків, пов'язаних з поєднанням IT і OT, і внутрішніх загроз.

Якщо в умовах вибору враховується можливість забезпечення безпеки, то основною проблемою є відсутність комплексних рішень кібербезпеки. Впровадження технології IIoT буде протистояти спробам злому до тих пір,

поки у нього не з'явиться надійна функція безпеки. Саме через це компанії, що займаються операційними технологіями, так обережні з технологією IIoT.

Проблеми аналітики

Навіть якщо рішення IIoT впроваджується на підприємстві, його фактична вартість повернення інвестицій реалізується за допомогою практичних даних, отриманих на основі зібраних даних IIoT. Це можливо реалізувати тільки за допомогою високопродуктивної аналітичної платформи, яка може обробляти великі обсяги даних.

При реалізації архітектури IIoT важливо, щоб аналіз даних також включав обробку, очищення та представлення даних. Це забезпечує достатній простір для можливого розширення, що дозволяє легко додавати аналітику в реальному часі або аналітичні передбачення в IIoT рішення [8].

Проблема збору і обробки даних

Пристрої, датчики та сенсори генерують і збирають величезну кількість даних, загальний обсяг зібраних даних може бути настільки великим, що може виявитися неможливим передати їх через мережу для подальшого аналізу. Наприклад один датчик зовнішньої температури на складі для виконання своєї ролі передає дані, включаючи температуру, вологість, рівень заряду батареї, версії програмного забезпечення, версії обладнання та зміни руху або положення.

Датчики можуть передавати цю інформацію кожні 30 секунд, і на складі може бути кілька сотень таких пристроїв. Це може бути тільки один з десятків типів датчиків, які мають різні типи даних, що в сукупності являє собою масивні неоднорідні набори інформації з багатьох джерел. Це призводить до великих витрат на збір даних та інтеграцію обладнання з системою обробки даних. Також великою проблемою є відсутність обчислювальної потужності і ресурсів зберігання для виконання складних завдань аналізу і машинного навчання.

Крім того, для великих об’ємів даних важко обробляти дані, так як для їх відповіді потрібно більше часу. Таким чином, передача конфіденційних даних через Інтернет для виконання важливого аналізу часто стає проблемою [9].

2.2. Аналіз існуючих рішень

Метод збору та обробки інформації за допомогою шлюзу

На рисунку 2.2 представлений метод, який містить три модулі: модуль інтелектуального об'єкта, модуль шлюзу і модуль центру керування (сервера). Кожен модуль є багаторівневим (включаючи сенсорний, мережевий і прикладний рівні) і виконує певні функції для підтримки моніторингу взаємозалежного середовища [10].

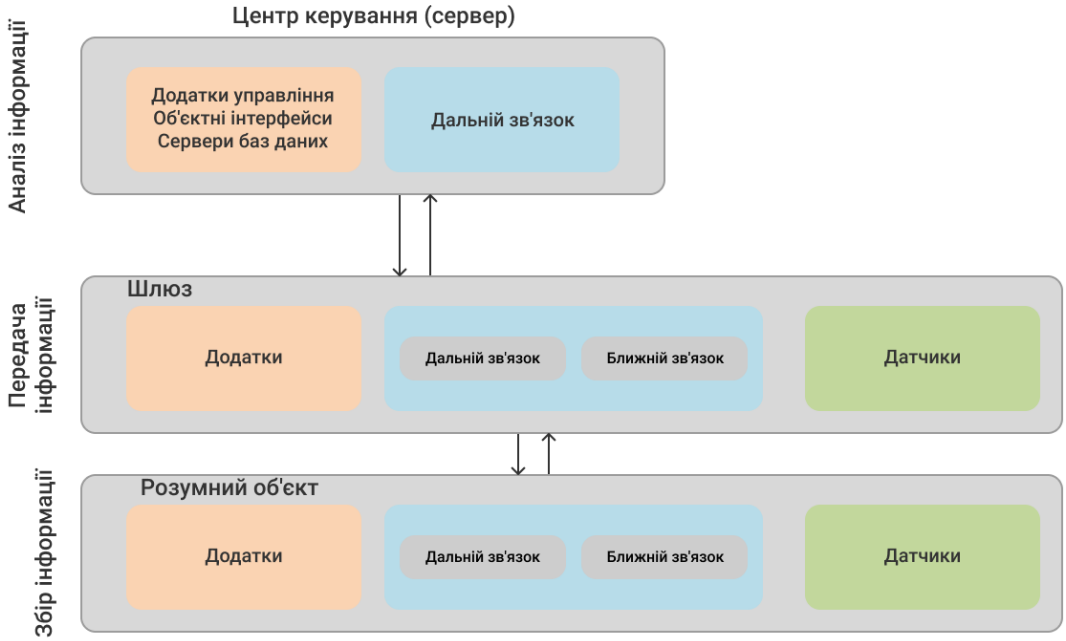

Рис. 2.2 Метод збору та обробки інформації за допомогою шлюза

Розумний об'єкт

Кожен смарт-об'єкт є фізичним пристроєм і безліч цих смарт-об'єктів розгортається на різних об'єктах (обладнанні). Смарт-об'єкти дозволяють сприймати, збирати дані і реагувати на конкретні умови. Таким чином дані,

зібрані цими інтелектуальними об'єктами, повинні бути доставлені на сервер для повної обробки і аналізу даних.

Модуль смарт-об'єкта складається з трьох рівнів: рівень сприйняття, мережевий рівень і прикладний рівень. Рівень сприйняття включає збір даних і взаємодію між смарт-об'єктами і шлюзами. Кожен смарт-об'єкт оснащений датчиками різних типів, такими як акустичні датчики, датчики температури, вологості, руху тощо.

Мережевий рівень відповідає за зв'язок між смарт-об'єктами, шлюзами і центром управління. Кожен інтелектуальний об'єкт може також включати радіопередавач на мережевому рівні для зв'язку на короткій відстані з іншими інтелектуальними об'єктами і шлюзами.

Шлюз

Модуль шлюзу є мостом між смарт-об'єктами і центром управління. З іншого боку, в разі, коли інтелектуальні об'єкти не мають прямого зв'язку, доступного для прямого зв'язку з центром управління за допомогою технологій телекомунікації; шлюзи будуть з'єднувати ці смарт-об'єкти з центром управління. Також модуль шлюзу буде виконувати обов'язки прикладного рівня при відсутності прикладного рівня в модулі смарт-об'єкта.

Безліч інтелектуальних об'єктів підключається до модуля шлюзу за допомогою зв'язку на короткій відстані. Рівень сприйняття в модулі шлюзу є необов'язковим, але кожен шлюз може також бути оснащений датчиками різних типів. Кожен шлюз може також включати радіопередавач на мережевому рівні для зв'язку на короткій відстані з іншими шлюзами і інтелектуальними об'єктами. Зв'язок між смарт-об'єктами і шлюзами може бути досягнутий за допомогою RPL (протокол маршрутизації для мереж з втратами) [11].

Програми, що працюють на смарт-об'єктах і модулях шлюзу, будуть виконувати дії в реальному часі. Наприклад оповіщення про пожежу,

відключення різного устаткування, евакуація персоналу і локалізація несправностей.

Центр керування

Модуль центру управління (сервера) відповідає за управління додатками і аналіз даних, зібраних від модулів смарт-об'єктів, генерування інформації та прийняття важливих рішень щодо аномальних подій, надаючи панель управління для підтримки процесу прийняття рішень.

Центр управління складається тільки з двох рівнів – мережевого і прикладного. Мережевий рівень відповідає за зв'язок між шлюзами, а в деяких випадках - безпосередньо зі смарт-об'єктами за допомогою технологій зв'язку на великі відстані.

Рівень додатків в основному відповідає за процеси управління і включає в себе інтерфейси, додатки IIoT, бази даних і API-інтерфейси служб, інструменти візуалізації тощо.

Коли зібрані дані проаналізовано, функції управління приводяться в дію без втручання людини. Центр управління виконує аналіз даних для двох основних цілей: для прогнозного обслуговування обладнання, аналізуючи дані про стан обладнання, що передаються інтелектуальними об'єктами в центр управління.

Таким чином, центр управління виконує профілактичне обслуговування для максимізації часу безвідмовної роботи і мінімізації збоїв, що дозволяє краще контролювати і обслуговувати різні активи зі зменшенням ризиків для здоров'я і безпеки. По-друге, центр управління аналізує дані про продуктивність виробництва.

Метод збору та обробки інформації з використанням розумного концентратора

На рисунку 2.3 наведений метод збору та обробки інформації, який дозволяє використовувати інформаційну та операційну інфраструктуру для

покращення управління, моніторингу та контролю існуючих і нових пристроїв. Він складається з трьох основних складових: рівень пристроїв, інтелектуальний концентратор, мікросервісна хмарна платформа. Представлений метод також може бути впроваджений на існуючих інтелектуальних підприємствах для оптимізації управління інтелектуальним обладнанням і для підвищення ефективності виробництва [12].

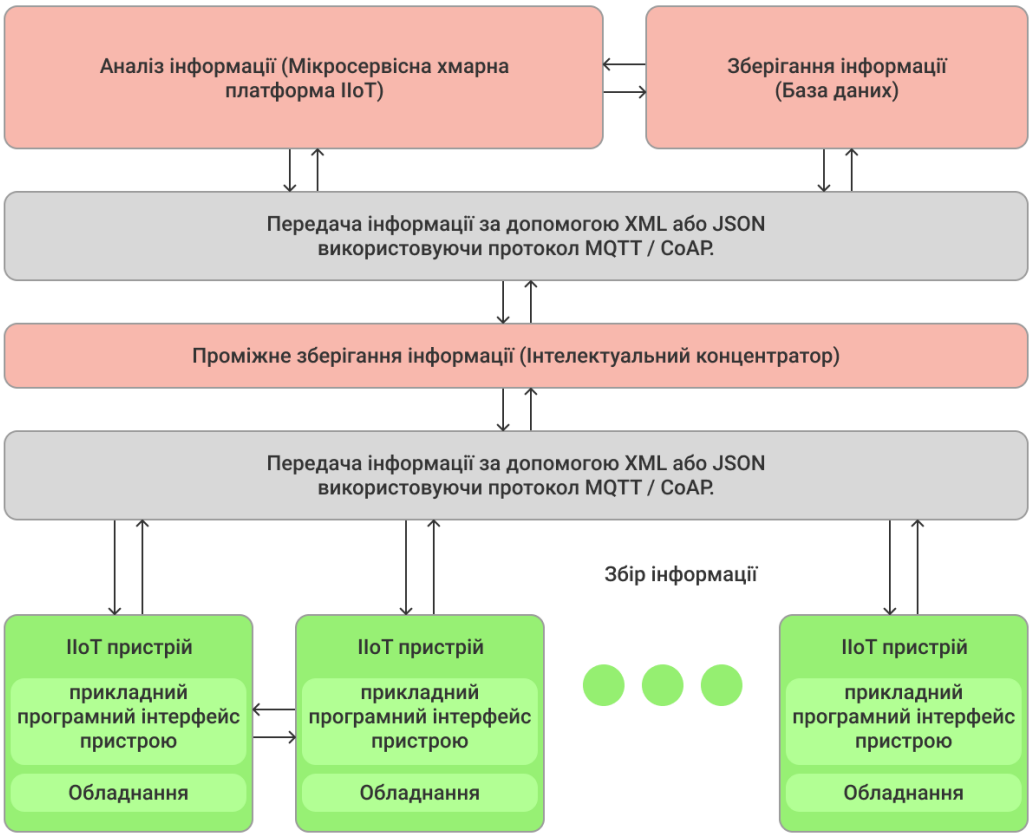

Рис. 2.3 Метод збору та обробки інформації з використанням розумного концентратора

Датчики

На цьому рівні пристрої оснащені різними датчиками, такими як температура, тиск, енергія, вібрація, швидкість обертання, сила, крутний момент, для моніторингу реального стану обладнання в цифровому вигляді. Всі фізичні пристрої в цеху інтегровані один з одним, і кожен пристрій має

унікальну ідентифікацію, так що будь-який користувач даних може отримати до них доступ, якщо має відповідні права доступу.

Інформація згенерована сенсорами, механізмами, пристроями, передається на інтелектуальний концентратор використовуючи протокол Інтернету речей CoAP або MQTT. Однак, оскільки різні пристрої можуть мати свої власні методи зв'язку, такі як Wi-Fi, Bluetooth / BLE, ZigBee і Z-Wave; неефективно безпосередньо підключати різні пристрої IoT до хмарної платформи, оскільки це може призвести до технічних труднощів, таким як затримка в мережі, великий обсяг трафіку, накладні витрати на зв'язок.

Зокрема, підключення датчиків до інфраструктури хмарних обчислень вимагає значних ресурсів, оскільки звернення до платформи буде здійснюватися кожен робочий цикл датчика. Найбільш прийнятним рішенням є збір даних з пристроїв IIoT з використанням їх методів зв'язку, а потім передача даних по дротовому зв'язку [13].

Інтелектуальний концентратор

Інтелектуальний концентратор розроблений для управління пристроями IIoT в розподілених місцях, включаючи інтелектуальні фабрики, склади і офіси. У практичних ситуаціях дротовий зв'язок є одним з надійних способів передачі даних на серверну платформу.

Інтелектуальний концентратор може діяти як шлюз для існуючих пристроїв IIoT. Функціональність концентратора складається з трьох функцій.

По-перше, він пропонує простий і зручний спосіб попередньої обробки даних, обміну даними та зв'язку між існуючими пристроями IIoT та серверною платформою, що означає, що при використанні інтелектуального концентратора можна оновлювати або вдосконалювати обмін даними та канали зв’язку.

По-друге, він забезпечує стійке рішення для вирішення завдань майбутнього розширення пристроїв IIoT.

По-третє, інтелектуальний концентратор призначений для створення безпечного зв'язку між пристроями IIoT та хмарною платформою IIoT на базі мікросервісів.

Хмарна платформа IIoT

Мікросервісна хмарна платформа IIoT виступає найголовнішим елементом методу. Вона складається з безлічі невеликих і незалежних сервісів. Кожен мікросервіс має свою власну бізнес-логіку і базу даних, і кожен сервіс може працювати самостійно. Мікросервіс може динамічно і гнучко розподіляти і використовувати ресурси для різних робочих завдань.

Архітектура мікросервісів має багато переваг перед традиційною монолітною архітектурою. З одного боку, використання різних пристроїв IIoT зростає експоненціально; важко передбачити кількість та різноманітність пристроїв IoT. Якщо використовується традиційна монолітна архітектурна платформа, всю платформу потрібно призупинити для обслуговування або модернізації, що негативно вплине на збір та обробку інформації.

З іншого боку, архітектура мікросервісів дозволяє користувачам додавати, видаляти, оновлювати або покращувати певний сервіс, не впливаючи на ефективність роботи всієї платформи, що вигідно для процесів обслуговування. За рахунок мікросервісів хмарна платформа IIoT може запропонувати єдину платформу з графічними інтерфейсами користувача (GUI) для промисловців, що дозволяє ефективно та зручно віддалено управляти, контролювати та керувати пристроями IIoT.

Метод збору та обробки інформації на основі промислових шин

На рисунку 2.4 представлений метод, який складається з чотирьох рівнів: рівень пристроїв, датчиків та механізмів, рівень постачальника даних, рівень міжплатформеного програмного забезпечення та рівня додатка [14].

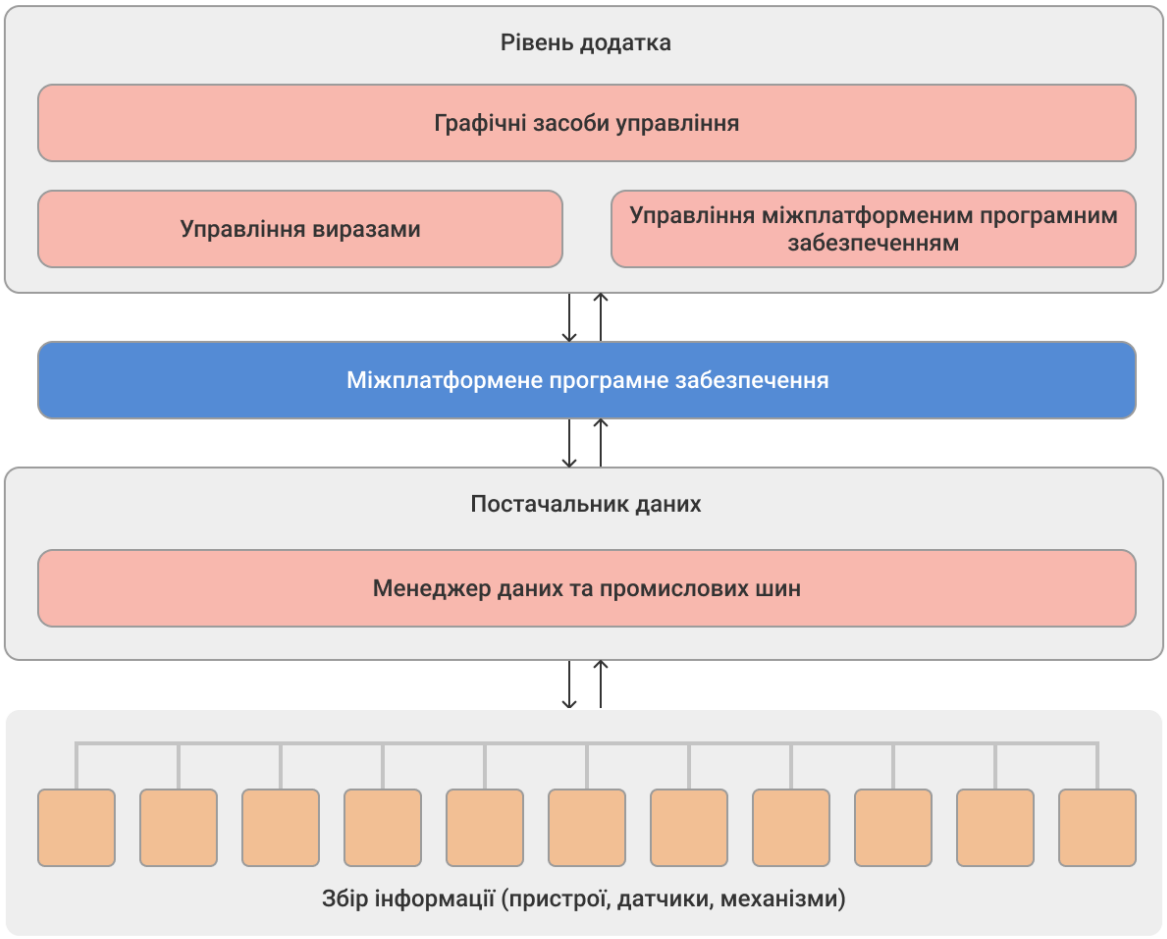

Рис. 2.4 Метод збору та обробки інформації на основі промислових шин

Рівень пристроїв

Рівень пристроїв складається з датчиків, механізмів, пристроїв для моніторингу та управління, які необхідно інтегрувати в архітектуру IIoT. Ці пристрої з'єднані між собою за допомогою дротових або бездротових промислових шин, які можуть працювати незалежно або зв’язуватися з іншими пристроями, підключеними до тієї ж промислової шини.

Важливим питанням при інтеграції методу збору та обробки інформації є вибір промислових шин, які мають свої особливості в умовах надійності та передбачуваності.

Крім того, для управління однією або кількома промисловими шинами можуть бути розроблені різні розумні пристрої. Для того, щоб відповідати вимогам промислового середовища (надійності та передбачуваності), ці

пристрої можуть бути розроблені на основі мікроконтролера та передавати дані на верхній рівень, використовуючи протокол зв'язку, наприклад MODBUS або ZigBee.

Одна або кілька промислових шин можуть бути підключені до комп'ютерів, які можуть бути географічно розподілені та обмінюватися даними через Інтернет.

Рівень постачальника даних

Цей рівень призначений для отримання даних з промислових шин та передачі їх на рівень міжплатформеного програмного забезпечення, який містить програмний комунікаційний модуль для кожної промислової шини.

Метою цього модуля є реалізація циклу збору інформації. Інтерфейс прийому і передачі даних використовується постачальником даних, окремим програмним модулем, для отримання даних з промислових мереж та для їхнього зберігання в кеш-пам'яті буферу. Ця пам'ять використовується для швидшого реагування на запити з рівня проміжного програмного забезпечення.

Розробка єдиного підходу для управління промисловими шинами може призвести до значного підвищення продуктивності. На верхньому рівні визначається інтерфейс API-DP, який використовується рівнем міжплатформеного програмного забезпечення для доступу до даних із промислових шин. Цей рівень пов'язаний зі списком промислових шин, з якими він може зв’язуватися, він також може зв’язуватися з однією або декількома промисловими шинами одного типу, причому кожна може мати свої параметрами зв'язку та різні порти зв'язку. Виходячи з цього списку створюються екземпляри програмних модулів-оболонок, кожна з яких має список пристроїв.

Ці файли опису використовуються модулями-оболонками для застосування буферної (кеш-пам’яті) пам’яті. На основі цієї інформації створюється структурований деревоподібний адресний простір, причому

перший рівень містить перелік промислових шин (кожна з унікальним описом), а другий рівень містить список пристроїв для кожної промислової шини.

Рівень міжплатформеного програмного забезпечення

Цей рівень забезпечує інтерфейс, який надає доступ до даних із промислових шин. Служба розповсюдження даних DDS дозволяє розробляти однорангові мережеві архітектури. Ще однією важливою особливістю цієї служби є можливість налаштування параметрів QoS, які дозволяють досягти необхідних показників у промислових процесах.

Рівень міжплатформеного програмного забезпечення представлений додатком, який отримує дані від постачальника даних та публікує їх через DDS. Елементи даних вибираються з адресного простору, наданого нижчим рівнем, і публікуються в домені як теми. Додаток може підписатися одну або більше тем для передачі даних на пристрої, які підключені до промислових шин.

Рівень додатка

На цьому рівні знаходиться додаток, який надає як графічну візуалізацію інформації, отриманої з промислових шин, так і передачу даних, яка буде надходити до пристроїв, підключених до промислових шин.

Цей додаток може створювати екземпляри трьох типів елементів управління: графічний, міжплатформеного програмного забезпечення і вирази. Кожен елемент управління має набір елементів даних, які можуть бути пов'язані між собою за допомогою математичного вираження.

Додаток цього рівня містить середовище публікація-підписка, яке використовується елементами управління як для публікації своїх елементів даних, так і для підключення їх до даних інших елементів управління, що публікуються в тому ж середовищі.

Порівняльна характеристика методів збору та обробки інформації

У таблиці 2.1 представлено порівняння основних технічних характеристик розглянутих методів збору та обробки інформації.

Таблиця 2.1

Зіставлення основних технічних характеристик

Метод	Протоколи та технології	Апаратні засоби	Інфраструктура
Метод збору та обробки інформації за допомогою шлюзу	Wi-Fi, Bluetooth, ZigBee	Розумні об’єкти, шлюз	Хмарні сервери
Метод збору та обробки інформації з використанням розумного концентратора	MQTT/ CoAP, Wi-Fi, BLE, ZigBee та Z-Wave	Інтелектуальний концентратор	Мікросервісна хмарна платформа
Метод збору та обробки інформації на основі промислових шин	CANOpen, ControlNet, Modbus	Шлюз	Хмарний сервер

Метод збору та обробки інформації з використанням розумного концентратора має перевагу в порівнянні з іншими методами, оскільки він допомагає в модернізації і досягненні високого рівня виробництва в обробній промисловості. Це досягається за рахунок того, що основним елементом цього методу виступає інтелектуальний концентратор, який надає зручні рішення при масштабуванні системи новими пристроями, виконує збір даних

та їх форматування, використовуючи процеси серіалізації, що дозволяє зменшити об’єми інформації, які необхідно передати на хмарну платформу для подальшого аналізу.

Висновки:

1) Проведено аналіз існуючих проблем IIoT, який показав наявність проблеми збору і обробки інформації і необхідність удосконалення цього процесу.
2) Проаналізовані існуючі методи збору та обробки інформації, на основі якого обрано прототип для подальшої модифікації.

РОЗДІЛ 3
МОДИФІКОВАНИЙ МЕТОД ЗБОРУ І ОБРОБКИ ІНФОРМАЦІЇ

3.1. Модифікований метод

На основі обраного прототипу модифікований метод збору та обробки інформації виглядає наступним чином (див. рис. 3.1).

Рис. 3.1 Модифікований метод

- Збір інформації

Основою збору даних є сенсори, датчики, механізми, які розосереджені на певній географічній області. Вони обмінюються даними по мережі для автономного збору і передачі даних на спеціальний вузол, який вважається проміжною точкою збору інформації.

- Серіалізація і десеріалізація

Для зберігання та передачі отриманої з пристроїв інформації використовується механізм серіалізації, що дозволяє зменшити розмір файлів, які передаються, що в свою чергу знизить навантаження на мережу і скоротить час, необхідний для передачі інформації для подальшого аналізу.

- Передача інформації

Отримана з сенсорів, механізмів, пристроїв інформація передається на

інтелектуальний концентратор, використовуючи протокол Інтернету речей CoAP або MQTT.

- Аналіз інформації та зберігання

Проаналізована на мікросервісній платформі IIoT, інформація може передаватися в інші інформаційні або операційні системи або зберігатися в базі даних.

3.2. Збір даних

В залежності від вимог до збору інформації передача зібраних даних може здійснюватися періодично або на основі подій. Інтелектуальний концентратор - це вузол з двома або більше мережевими інтерфейсами, він збирає і виконує проміжну обробку даних, які надійшли з пристроїв збору інформації.

Сенсори, датчики і механізми являють собою невеликі електронні пристрої, здатні вимірювати фізичні величини (наприклад, температуру, світло, тиск) і передавати її на вузол обробки інформації. З огляду на досягнення в галузі мікроелектроніки, технології і програмне забезпечення для бездротової передачі даних дозволяють виробляти мікродатчики обсягом кілька кубічних міліметрів [14].

Загальний процес збору інформації та передачі на інтелектуальний концентратор представлений на рисунку 3.2. Пристрої можуть мати свої методи зв'язку, наприклад, Wi-Fi, BLE, ZigBee и Z-Wave.

Рис. 3.2 Загальний процес збору інформації

Технології для передачі даних на інтелектуальний концентратор

Сенсорні мережі - це просторово розподілені датчики, які відстежують фізичні або екологічні умови, такі як температура, звук, тиск тощо і спільно передають свої дані через мережу. Такі мережі можуть використовуватися у великій кількості різноманітних застосувань: промислова автоматизація, системи контролю мікроклімату, системи безпеки та пожежної сигналізації, облік споживання енергії та її оптимізація тощо [16].

Площа покриття таких мереж може становити від декількох метрів до кількох кілометрів. Одними з головних стандартів впровадження цих мереж є Wi-Fi, BLE, ZigBee, Z-Wave.

Застосування технології Wi-Fi в модифікованому методі

Для реалізації модифікованого методу можливе використання Wi-Fi. Основою мережі є точка доступу (AP). Основним завданням точки доступу є передача бездротового сигналу. Для підключення до точки доступу та підключення до бездротової мережі пристрої повинні бути оснащені адаптерами бездротового мережі.

Точка доступу надсилає ідентифікатор мережі, використовуючи спеціальні пакети. Швидкість передачі цих пакетів 0,1 Мбіт/с кожні 0,1 с.

Клієнт, знаючи ідентифікатор мережі, перевіряє можливість підключення до даної точки доступу. Якщо в зоні дії дві точки доступу з однаковими ідентифікаторами мережі, приймач має можливість вибрати між ними, орієнтуючись на рівень сигналу.

Топологія мережі з об’єднаними точками доступу в єдину мережу під управлінням контролера представлена на рисунку 3.3.

Рис. 3.3 Централізована мережа Wi-Fi для збору інформації на інтелектуальний концентратор

BLE для зменшення енергоспоживання в модифікованому методі

Бездротова технологія BLE в модифікованому методі може використовуватися для мінімізації швидкісті з’єднання з пристроями. На відміну від класичного Bluetooth, BLE постійно знаходиться в сплячому режимі, за винятком випадків, коли встановлюється з'єднання, що зменшує енергоспоживання в режимі простою.

Фактичний час з'єднання становить всього кілька мілісекунд. Причина, по якій з'єднання займають так мало часу в тому, що швидкість передачі даних дуже висока - 1 Мбіт/с.

Сітчаста топологія використовується при наявності декількох ведучих вузлів BLE. (див. рис. 3.4). У цій топології один вузол функціонує як

центральний і діє як головний, а всі інші вузли функціонують як ведучі та підлеглі вузли.

Рис. 3.4 Сітчаста топологія BLE для збору інформації в модифікованому методі

Технологія ZigBee для збору інформації

У модифікованому методі можна використовувати сітчасту топологію ZigBee (див. рис. 3.5), оскільки надмірність є важливим фактором в промисловості.

Швидкість передачі даних 250 кбіт / с підходить для періодичної і проміжної двосторонньої передачі даних між датчиками і координатором.

У сітчастій топології мережа Zigbee розширюється декількома маршрутизаторами, де координатор відповідає за їх запуск. Ці структури дозволяють будь-якому пристрою зв'язуватися з будь-яким іншим сусіднім вузлом для забезпечення надмірності даних. Якщо який-небудь вузол виходить з ладу, інформація автоматично направляється на інший пристрій.

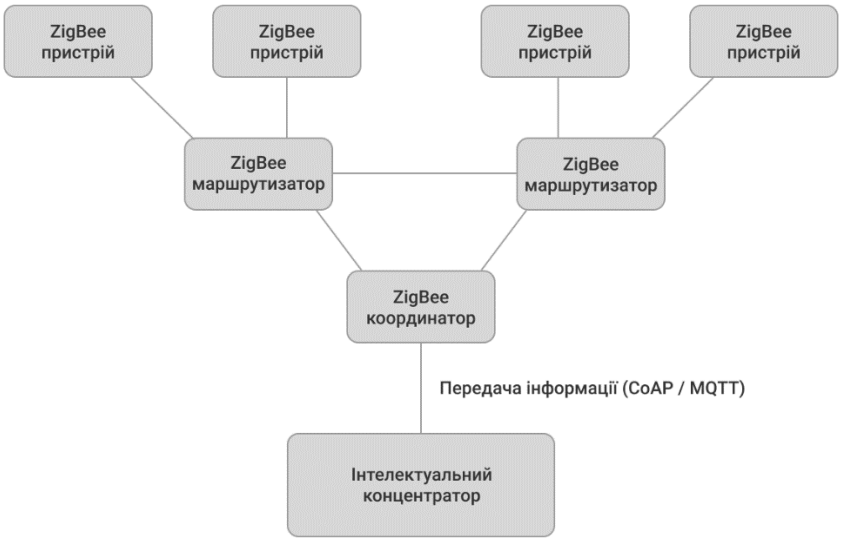

Рис. 3.5 Сітчаста топологія ZigBee з маршрутизацією для збору інформації

Застосування Z-Wave в межах модифікованого методу

При використанні Z-Wave в модифікованому методі збору і обробки інформації, мережа складається з контролерів і підлеглих пристроїв (див. рис. 3.6).

Рекомендується мати пристрій Z-Wave приблизно кожні 10 метрів або ближче для максимальної ефективності [17].

Контролер, який створює мережу Z-Wave є основним контролером. Цей первинний контролер являється головним контролером у мережі. Основний контролер має можливість включати і виключати вузли мережі, а також піклується про управління розподілом ідентифікаторів вузлів.

Контролери, які додаються в мережу Z-Wave з використанням основного контролера, називаються вторинними контролерами. Підлеглі пристрої - це вузли, які відповідають на основі отриманих команд, а також виконують команди. Підлеглі вузли також направляють команди іншим вузлам в мережі.

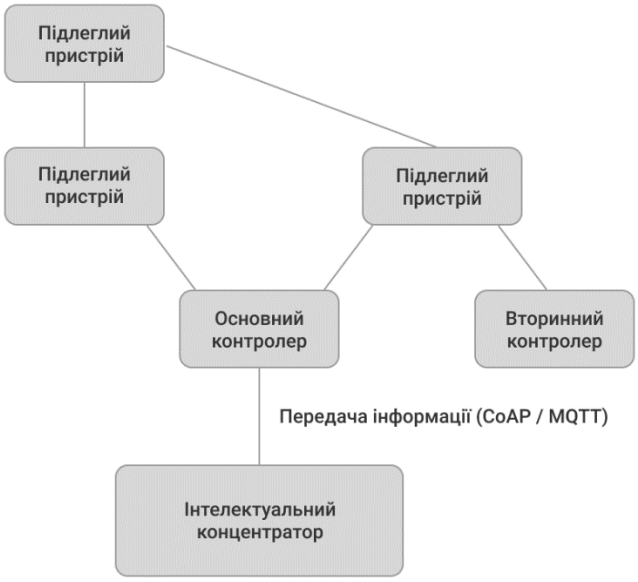

Рис. 3.6 Приклад топології Z-Wave в межах модифікованого методу

3.3. Протоколи для передачі інформації

CoAP

В модифікованому методі цей протокол призначений для передачі між пристроями в одній обмеженій мережі, наприклад, з низьким енергоспоживанням, в мережах з втратами, а також між пристроями в різних мережах, з'єднаних Інтернетом.

CoAP організований в два рівня:

- Рівень повідомлень;
- Рівень запит / відповідь.

На рисунку 3.7 показана структура CoAP.

Рис. 3.7 Структура CoAP для передачі інформації в модифікованому методі

Рівень повідомлень

Рівень повідомлень обробляє єдиний обмін повідомленнями між кінцевими точками. Повідомлення обміну на цьому рівні можуть бути чотирьох типів:

- Confirmable - вимагає підтвердження;
- Non-confirmable - не вимагає підтвердженням;
- Квитування - підтверджує отримання Confirmable повідомлення;
- Reset - вказує на те що Confirmable повідомлення було отримано, але контекст, що підлягає обробці, відсутній.

Даними повідомленнями CoAP забезпечує механізм власної надійності. Рівень повідомлень також забезпечує підтримку багатоадресної передачі.

Рівень запит / відповідь

CoAP підтримує URI (універсальний ідентифікатор ресурсу), що дозволяє одержувачеві визначати повноваження, шлях і ідентифікацію ресурсу.

CoAP підтримує наступні методи:

1) GET. Метод GET використовується для отримання будь-якої інформації, ідентифікованої URI-запиту;
2) POST. Метод POST служить для запитів, при яких сервер приймає дані, поміщені в тіло повідомлень, для зберігання;
3) PUT. Метод PUT запитує, щоб ресурс, ідентифікований URI-запиту, був оновлений або створений з вкладеним тілом повідомлення. Якщо ресурс існує по цьому URI, тіло повідомлення буде вважатися модифікованою версією цього ресурсу. Якщо ресурс не існує, сервер може створити новий ресурс з цим URI. Якщо ресурс не може бути створений або змінений, тоді слід відправити відповідний код помилки;
4) DELETE. Метод DELETE запитує видалення ресурсу, ідентифікованого URI-запиту.

С точки зору безпеки, протокол CoAP побудований з DTLS, який забезпечує ті ж гарантії, що й TLS, але він працює поверх UDP.

CoAP може бути інтегрований з такими форматами даних, як XML, JSON, Protobuf для ефективного зв'язку з іншими платформами.

MQTT

В модифікованому методі цей протокол забезпечує мінімальні вимоги до ресурсів та використовується для передачі інформації.

MQTT працює з моделлю клієнт-сервер де кожен контролер є клієнтом і з’єднаний з інтелектуальним концентратором, який є брокером. Наявність брокера обов'язково, оскільки він керує розподілом даних підписникам. Всі контролери посилають дані тільки брокеру і приймають дані теж тільки від нього [18].

На рисунку 3.8 показана структура роботи протоколу MQTT.

Рис. 3.8 Структура роботи протоколу MQTT для передачі інформації в модифікованому методі

У мережі на базі протоколу MQTT розрізняють 3 об'єкти:

1) Видавець - MQTT-клієнт, який при виникненні певної події передає брокеру інформацію про неї, публікуючи відповідні топіки (теми);
2) Брокер - MQTT-сервер, який приймає інформацію від видавців і передає її відповідним підписникам. Різні брокери можуть з'єднуватися між собою, якщо вони підписуються на повідомлення один одного;

3) Підписник - MQTT-клієнт який після підписки на брокера більшу частину часу "слухає" його і постійно готовий до прийому і обробки вхідного повідомлення.

Коли видавець відправляє дані брокеру, це називається «публікацією». Коли підписник хоче отримати дані від брокера, він «підписується» на тему. В подальшому він буде отримувати всі повідомлення, опубліковані на цю тему.

Видавець також відправляє з самим повідомленням рівень QoS (якість обслуговування). Цей рівень визначає гарантію доставки повідомлення [19].

Ці рівні QoS наступні:

1) QoS0 - повідомлення передається тільки один раз і не вимагає підтвердження. Цей рівень не повинен використовуватися для критично важливої інформації, так як існує ризик того, що підписники не отримають повідомлення;
2) QoS1 - повідомлення відправляється мінімум один раз і вимагає підтвердження від брокера щодо конкретного повідомлення;
3) QoS2 - видавець і брокер працюють разом, щоб гарантувати, що брокер отримає і відреагує на повідомлення рівно один раз. Це вимагає додаткових витрат оскільки використовується механізм чотирьохетапного підтвердження пакетів. Хоча це найбезпечніший рівень QoS, він також самий повільний і тому використовується тільки при необхідності.

Більшість MQTT-брокерів підтримує використання TLS, що дозволяє додаткам захистити обмін конфіденційними даними. Стандартний порт для захищеного MQTT-з'єднання має номер 8883 [20].

Протокол не вводить обмежень на формат даних.

3.4. Серіалізація

Серіалізація – це процес перетворення структур даних або стану об’єкта в послідовність бітів, який може бути відновлений пізніше. Коли

результуюча послідовність бітів перечитується відповідно до формату серіалізації, вона може використовуватися для створення ідентичного клону вихідного об’єкта. Відновлення вихідного стану структури даних із послідовності бітів називається операцією десеріалізації [21].

На рисунку 3.9 зображено загальний процес серіалізації-десеріалізації.

Рис. 3.9 Загальний процес серіалізації-десеріалізації

При необхідності створення розподіленого сервісу, частини якого мають обмінюватися інформацією зі складною структурою, у цьому випадку для даних, які планується передавати, створюється код, який виконує процеси серіалізації та десеріалізації. Для об’єкта, заповненого необхідними даними, викликається створений код серіалізації, в результаті на виході отримуємо, наприклад, XML файл.

Отримана в результаті серіалізації послідовність бітів записується у базу даних, пам’ять або файл, який потім надсилається приймальній стороні. Для десеріалізації сервіс-одержувач створює об’єкт такого ж типу і викликає необхідний код, в результаті чого отримуємо об’єкт з такими ж даними, які мав об’єкт сервіса-відправника.

JSON

JSON в основному використовується для передачі серіалізованих даних по мережевому з'єднанню. Він може передавати дані між двома комп'ютерами, базою даних, програмами тощо.

JSON описує одну з двох структур даних:

- Набір пар «ключ:значення». Ключем може бути тільки рядок, а значенням будь-яка форма JSON, пари між собою не впорядковані;
- Упорядкований список. Елементами можуть бути будь-які форми JSON.

В якості значень JSON можуть бути використані:

- Об’єкт – це множина пар ключ:значення, укладена у фігурні дужки. Пари ключ:значення відділяються одна від одної комами;
- Масив – це впорядкована множина значень. Масив поміщається в квадратні дужки, всередині яких значення розділяються комами. Масив може бути порожнім, тобто не містити жодного значення;
- Число - як ціле, так і з плаваючою комою;
- Літерали – записи, що представляє собою фіксоване значення. Зазвичай виділяють наступні елементарні типи літералів: числові, строкові, логічні, пусте значення (null);
- Рядок – це впорядкована множина з нуля або більше символів юнікода, яка обмежена подвійними лапками. Символи можуть бути вказані з використанням керуючих послідовностей, які починаються зі зворотної риски (підтримуються варіанти \n, \ r, \ f та \b).

XML

В модифікованому методі XML можна використовувати для обміну інформацією між інтелектуальним концентратором та мікросервісною платформою. Також розширювану мову розмітки можна використовувати для зберігання та впорядкування даних, що дозволяє налаштувати процес обробки даних.

XML визначає набір правил для кодування документів в форматі і він може бути прочитаний людиною і машиною. Розмітка - це інформація, що додається до документа, яка певним чином посилює його значення, оскільки вона ідентифікує його частини і то, як вони пов'язані одна з одною. Мова

розмітки являє собою набір символів, які можуть бути поміщені в текст документа для розмежування і маркування частин цього документа.

Існують дві важливі характеристики XML:

- XML являється розширюваним. Він дозволяє створювати власні теги, які підходять вашому додатку;
- XML переносить дані, не представляє їх. Він дозволяє зберігати дані незалежно від того, як вони будуть представлені.

Синтаксис XML

На рисунку 3.10 наведені правила синтаксису для запису різних типів розмітки і тексту в XML-документі.

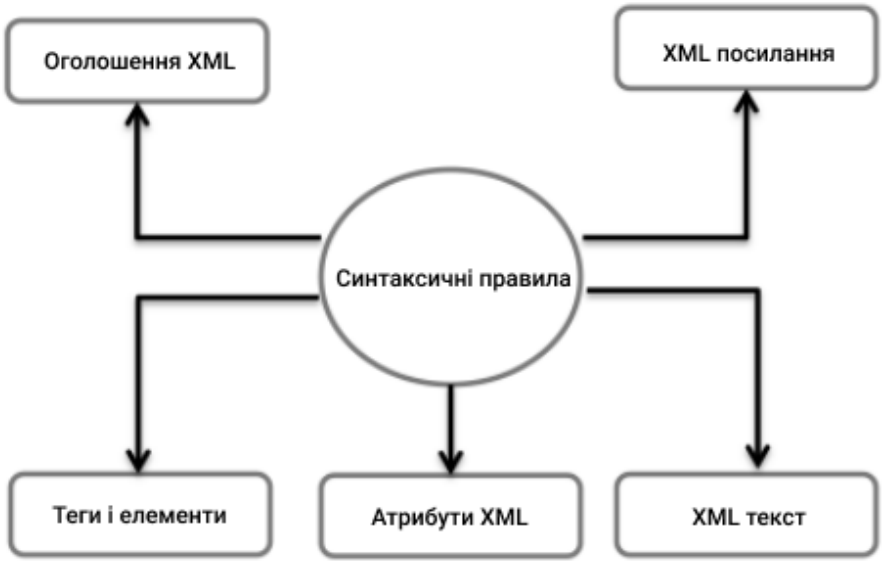

Рис. 3.10 Правила синтаксису XML

Оголошення XML

XML документ може додатково мати оголошення XML. Синтаксичні правила для оголошення XML:

- Оголошення XML враховує регістр і написано малими літерами;
- Оголошення XML строго повинно бути першою інструкцією в XML-документі.

Теги і елементи

Файл XML структурований декількома XML-елементами, які також називаються XML-вузлами або XML-тегами, імена XML-елементів укладені в трикутні дужки. Кожен XML-елемент повинен бути закритий або початковими, або кінцевими елементами.

Вкладеність елементів: XML-елемент може містити кілька XML-елементів в якості своїх дочірніх елементів, але дочірні елементи не повинні перекриватися. Тобто кінцевий тег елемента повинен мати те ж ім'я, що і останній неспівпадаючий початковий тег.

Атрибути XML

Атрибут задає одну властивість для елемента, використовуючи пару ім'я-значення. Елемент XML може мати один або кілька атрибутів.

XML посилання

Посилання зазвичай дозволяють додавати або включати додатковий текст або розмітку в документ XML. Посилання завжди починаються з символу «&», який є зарезервованим символом, і закінчуються символом «;». XML має два типи посилань:

- Посилання на сутності. Посилання на сутність містить ім'я між початковим і кінцевим роздільниками. Наприклад, &example, де example - це ім'я. Ім'я відноситься до попередньо визначеного рядку тексту або розмітки;
- Посилання на символи. Вони містять посилання, такі як A містять хеш-знак «#», за яким слідує число. У цьому випадку 65 відноситься до букви алфавіту "A".

XML текст

Імена XML-елементів та XML-атрибутів чутливі до регістру, що означає, що імена початкових і кінцевих елементів повинні бути записані в

одному і тому ж регістрі. Щоб уникнути проблем з кодуванням символів, всі файли XML слід зберігати як файли Unicode UTF-8 або UTF-16. Розділові символи, такі як пробіли, табуляції і розриви рядків між XML-елементами і між XML-атрибутами будуть ігноруватися. Деякі символи зарезервовані самим синтаксисом XML.

Protobuf

Protobuf - це гнучкий, ефективний, автоматизований механізм для серіалізації даних. Для використання цього механізму необхідно визначити, як дані повинні бути структуровані один раз, а потім ви можете використовувати спеціальний згенерований вихідний код, щоб легко записувати і зчитувати структуровані дані, використовуючи різні мови програмування. Protobuf дозволяє оновити свою структуру даних, не порушуючи розгорнуті програми, скомпільовані зі «старим» форматом [22].

Щоб визначити форму повідомлення, користувач надає дескриптор повідомлення у файлі «.proto», який передається компілятору Protobuf.

Імена полів використовуються тільки користувачами для доступу або оновлення цих полів і не відображаються в серіалізованих даних. Теги повідомлення повинні бути унікальними числами, і вони використовуються для ідентифікації полів в закодованому двійковому форматі.

З цього описання компілятор Protobuf згенерує реалізацію типу даних на обрану мову програмування для управління, серіалізації і десеріалізації повідомлень.

Структуровані дані

Protobuf підтримує більше типів, ніж просто int32 і int64, включаючи числа з плаваючою комою, логічні значення і рядки. Крім того, типи можуть бути скалярними або повторними.

Скалярні типи включають базові типи, такі як цілі числа, зіставні типи, такі як перерахування. Так само можуть бути типи, які визначаються

користувачем, вони зазвичай називаються вбудованими повідомленнями або вбудованими полями.

Повторні типи - це просто послідовності деякого скалярного типу, включаючи вбудовані повідомлення.

Важливо відзначити, що кожен тип має значення за замовчуванням, яке використовується, коли поле не включене в закодоване повідомлення. Використання значень за замовчуванням може зменшити розмір повідомлень, оскільки немає необхідності кодувати поле зі значенням за замовчуванням.

Серіалізовані дані

Закодоване повідомлення Protobuf представляється двійковим рядком, який, по суті, являє собою послідовність пар ключ-значення. Ключ для кожної пари в повідомленні має два значення - номер поля з вашого .proto файлу і його тип, який надає достатньо інформації, щоб знайти довжину наступного значення.

Як приклад, повідомлення Timestamp, поле seconds якого дорівнює 1, а поле nanos дорівнює 10, може бути закодовано у вигляді двійкового рядка: 08 01 10 0A. Перший байт, 08, являє собою тег і тип поля. У першому наближенні цей байт означає, що наступний байт 01 є цілим числом і має тег поля 1, тобто це значення поля seconds. Точно так же третій байт, 10, вказує, що наступний байт є значенням поля nanos.

У таблиці 3.1 наведені доступні типи поточної версії Protocol Buffer.

Таблиця 3.1

Доступні типи поточної версії Protobuf

Тип	Значення	Використання
0	Variant	int32, int64, uint32, uint64, sint32, sint64, bool, enum
1	64-bit	fixed64, sfixed64, double

Продовження таблиці 3.1

Тип	Значення	Використання
2	Length-delimited	string, bytes, embedded messages, packed repeated fields
3	Start group	groups (deprecated)
4	End group	groups (deprecated)
5	32-bit	fixed32, sfixed32, float

Формат включає тип поля як частину ключа, щоб кожна пара містила достатню інформацію для визначення довжини її значення. Кожен тип пов'язаний з окремим номером і визначає, як повинні бути десеріалізовані наступні байти.

У цьому форматі сім бітів кожного закодованого байта представляють відповідні сім бітів цілого числа, в той час як найстарший біт вказує, чи повинні бути включені наступні байти.

Тег поля і його тип упаковані разом в ціле число, причому три молодших біта кодують тип значення, а старші біти кодують тег поля. Приклад показаний на рисунку 3.11.

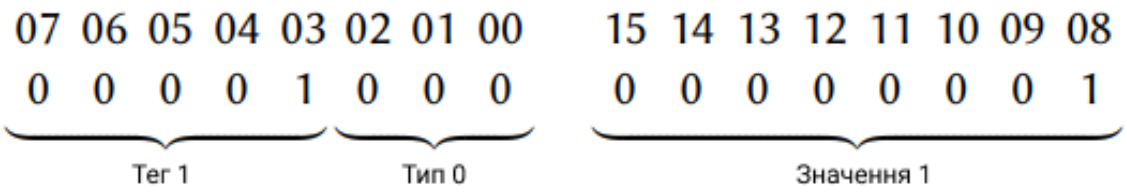

Рис. 3.11 Кодовані біти, 08 01, для поля seconds

При серіалізації типу з роздільником по довжині спочатку кодується кількість байтів закодованого значення як varint, а потім фактичне значення.

Порівняння методів серіалізації

У таблиці 3.2 представлений опис основних характеристик методів серіалізації з використанням XML, JSON, Protobuf.

Таблиця 3.2

Опис основних характеристик методів серіалізації

Характеристики	JSON	XML	ProtoBuf
Стандартизація	Так	Так	Так
Специфікація	STD 90 / RFC 8259	Рекомендації W3C : 1.0 и 1.1	Developer Guide: Encoding
Бінарний	Ні	Ні	Ні
Доступний для читання людиною	Так	Так	Ні
Підтримка посилань	Так	Так	Ні
Стандартні API	JSONQuery, JSONPath, JSON-LD	DOM, SAX, XQuery, XPath	C++, Java, C#, Python, Go, Ruby, Objective-C, C, Kotlin та інші.

Для модифікованого методу збору та обробки інформації пропонується використовувати метод серіалізації Protobuf.

В порівнянні з Protobuf, JSON і XML передають дані з деталями метаданих, що додає навантаження на корисне навантаження. Використовуючи Protobuf для серіалізації та десеріалізації буде витрачено менше процесорного часу та пам'яті, тому час обробки швидше порівняно з JSON та XML.

Protobuf стискає дані та генерує щільні дані. Якщо його порівнювати з XML, він займає майже 1/3 розміру, а якщо порівнювати з JSON, то 1/2 [23].

JSON та XML доступні для читання людиною та не захищені для передачі даних по мережі. Якщо необхідно, щоб відповідь не читалася користувачем, тоді необхідно використовувати Protobuf. Також користувачеві потрібен файл .proto для десеріалізації об'єктного потоку.

3.5. Мікросервісна платформа

В модифікованому методі мікросервісна платформа структурує додаток як набір сервісів. Мікросервіси, як правило, засновані на бізнес-функціях.

Розгортання з використанням контейнерів забезпечує переносимість. Контейнеризація скорочує час розгортання, оскільки вимагає лише включення необхідного контейнера, не зачіпаючи інші контейнери, які працюють на тому ж хості [24].

Кожен мікросервіс створюється з використанням мови програмування, таких як Python, Ruby, C ++ або Java, в залежності від бізнес-функції.

Контейнери забезпечують ізоляцію при роботі на одному і тому ж хості, що робить розгортання цих різноманітних сервісів більш швидким і простим. За допомогою мікросервісів користувачі можуть легко створювати і підтримувати великі і складні додатки.

IIoT має величезну екосистему розгортання, що містить безліч серверів, додатків, датчиків і протоколів. Він також має безліч кінцевих точок, таких як програмне забезпечення та контроль якості, які вимагають інтенсивної інтеграції між пристроєм, даними і додатками.

У випадку мікросервісів функціональність розбивається на компоненти найнижчого рівня у вигляді невеликих, модульних, незалежно розгорнутих і слабо пов'язаних служб, що знижує складність інтеграції, з якою стикається монолітна архітектура.

Мікросервісна платформа може передавати корисні статистичні дані або інформацію в інші інформаційні або операційні системи, такі як система управління виробництвом, автоматизована система зберігання і пошуку і система управління активами підприємства за допомогою Protobuf, використовуючи протокол CoAP або MQTT.

Висновки:

1) В цьому розділі запропоновано модифікований метод збору та обробки інформації, який відрізняється використанням методу Protobuf для серіалізації інформації.
2) Модифікований метод дозволяє зменшити час, необхідний для обробки інформації та зменшити об’єм інформації, яка передається по мережі за рахунок використання серіалізації.

РОЗДІЛ 4
МОДЕЛЮВАННЯ ТА ПОРІВНЯЛЬНА ОЦІНКА ЗАПРОПОНОВАНОГО РІШЕННЯ

4.1. Натурне моделювання модифікованого методу

В процесі моделювання виконується серіалізація файлів перед їх відправкою через Інтернет, оскільки це може значно скоротити час, необхідний для цього. Серіалізація інформації також знижує фінансові витрати на експлуатацію мережі, оскільки для передачі даних потрібно менше обладнання та менша смуга пропускання. Також серіалізовані файли зменшують обсяг простору, необхідний для зберігання даних [25, 26].

Для отримання порівняльної оцінки запропонованого рішення досліджено три методи серіалізації з використанням XML, JSON, Protobuf за наступними критеріями (див. рис. 4.1):

- час, необхідний для серіалізації вхідних даних;
- час, необхідний для десеріалізації серіалізованих даних;
- час, необхідний для серіалізації та десеріалізації вхідних даних;
- розмір серіалізованого файлу.

За обраними критеріями можна визначити наскільки знизиться час та швидкість обробки інформації, необхідні для відправки інформації, використовуючи запропонований метод.

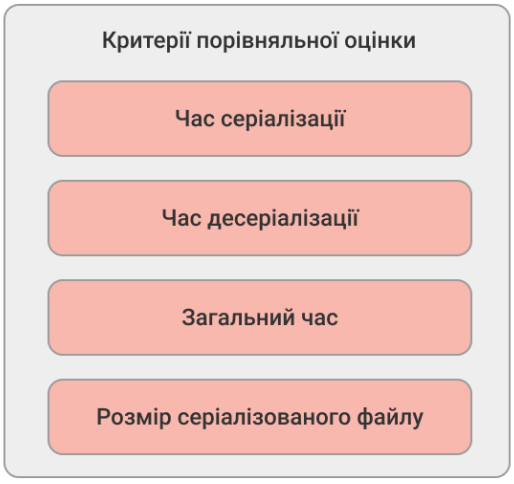

Рис. 4.1 Критерії порівняльної оцінки

На рисунку 4.2 наведена натурна модель. Згенерована пристроєм інформація передається на контролер. На контролері викликається процес серіалізації, в результаті якого, отримуємо серіалізований файл, який можна передавати по мережі. На стороні приймача викликається процес десеріалізації і відновлюється початкова інформація.

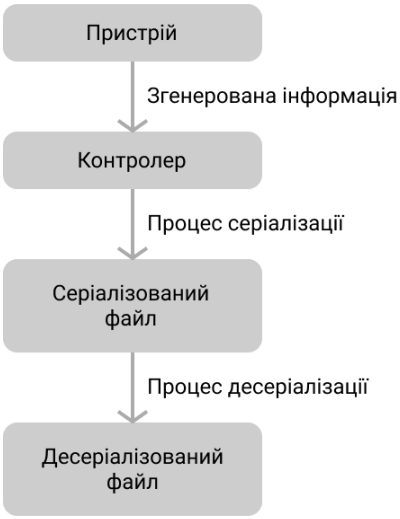

Рис. 4.2 Натурна модель

Моделювання проводилося на комп’ютері з наступними характеристиками:

1) Процесор: Intel(R) Core(TM) i5-8250I CPU @ 1.60 ГГц 1.80 ГГц;
2) Оперативна пам'ять: 8,00 ГБ;
3) Запам'ятовувальний пристрій: SSD;
4) Тип системи: 64-розрядна операційна система, процесор x64;
5) Операційна система: Windows 10.

Випробування повторюються для різної кількості вхідних даних (див. табл. 4.1). Ініціалізуємо експеримент для запису в файл різними методами серіалізаці 50, 500, 5000, 5000 рядків вхідного файлу. У кожному випробуванні записуємо відповідну кількість рядків згенерованих даних у файл. Для кожного методу (XML, JSON, Protobuf) проводимо операції серіалізації і десеріалізації 9 разів і розраховуємо середнє значення для кожного методу. В кожній із 9 ітерації для кожного методу вимірюємо час

серіалізації, десеріалізації та загальний час в мілісекундах. Також вимірюємо розмір серіалізованого файлу для кожного методу. На рисунку 4.3 наведений процес проведення моделювання

Для серіалізації у випробуваннях використовується клас Object, який складається з типів int, double, string, а також списку об'єктів. Результати випробувань зберігаються в текстові файли.

Таблиця 4.1

Вхідні параметри для моделювання

Типи даних	int, double, string, список об’єктів
Кількість вхідних даних (рядків)	50, 500, 5000, 50000
Кількість випробувань	4
Кількість ітерацій для кожного випробування	9

Рис. 4.3 Процес моделювання

Результати завжди будуть відрізнятися для різних апаратних засобів, операційних систем, але є те, що залишається незмінним - відносна різниця між методами серіалізаціїї.

Перше випробування

У першому випробувані вхідні дані складаються з 50 рядків. Після кожної ітерації фіксуємо час, який необхідний для серіалізації, десеріалізації, загальний час та розмір файлу (див. рис. 4.4). У таблиці 4.2 записані середні значення для першого випробування.

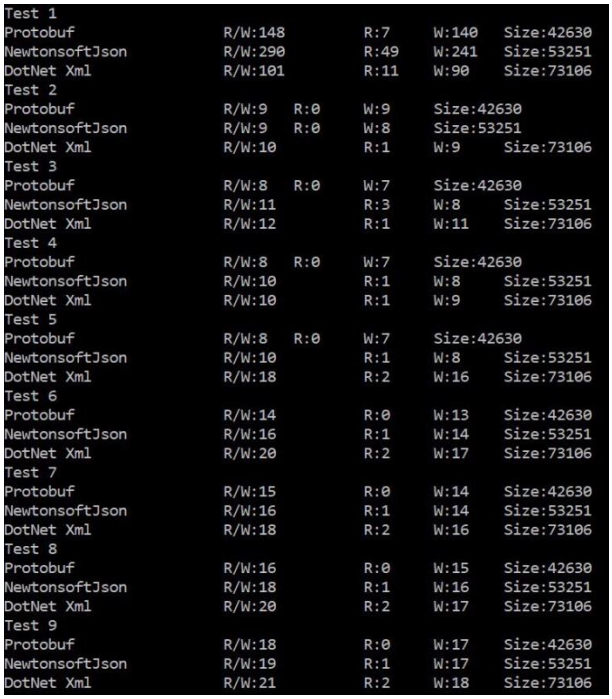

Рис. 4.4 Результати першого випробування

Таблиця 4.2

Усереднені результати першого випробування

Метод	Час серіалізації, мс	Час десеріалізації, мс	Загальний час, мс	Розмір файлу, байт
Protobuf	25,4	0,7	27,1	42630
JSON	37,1	6,4	44,3	53251
XML	25,6	2,7	28,5	73106

Як видно з таблиці, для першого випробування найкращий результат отримано використовуючи метод Protobuf, хоча загальний час серіалізації та десеріалізації майже не відрізняється від XML. З іншого боку, розмір XML у випробуванні становить 73106 байт, що більше ніж у Protobuf та JSON.

Друге випробування

У другому випробувані вхідні дані складаються з 500 рядків. Після кожної ітерації, як і у попередньому випробуванні, фіксуємо час, який необхідний для серіалізації, десеріалізації, загальний час та розмір файлу (див. рис. 4.5). У таблиці 4.3 записані середні значення для другого випробування.

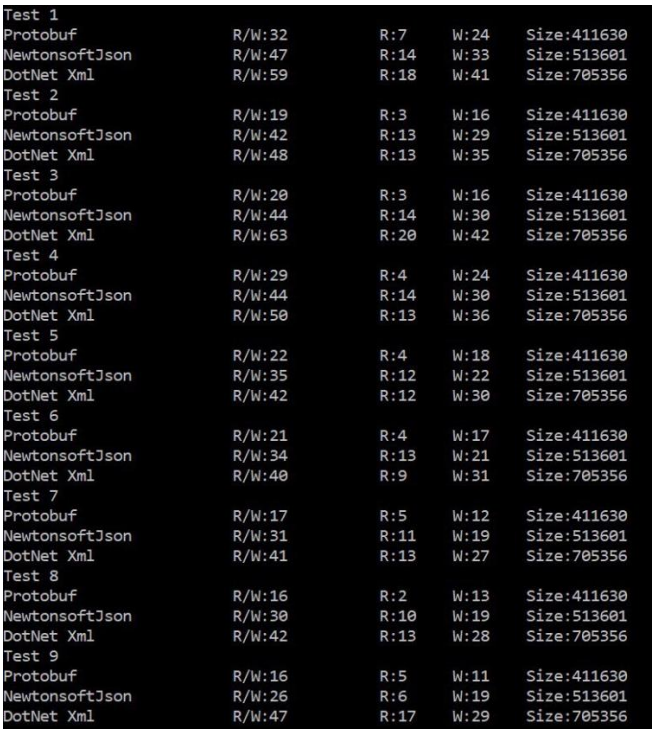

Рис. 4.5 Результати другого випробування

Таблиця 4.3

Усереднені результати другого випробування

Метод	Час серіалізації, мс	Час десеріалізації, мс	Загальний час, мс	Розмір файлу, байт
Protobuf	16,8	4,1	21,3	411630
JSON	24,7	11,9	37	513601
XML	33,2	14,2	48	705356

В результаті другого випробування найкращі результати отримано використовуючи Protobuf, проте різниця між Protobuf та XML змінилася

майже в 2 рази. Як і у попередньому випробувані, найбільший розмір серіалізованого файлу отримано використовуючи XML.

Третє випробування

У третьому випробувані вхідні дані складаються з 5000 строчок. Після кожної ітерації, як і у попередніх випробуваннях, фіксуємо час, який необхідний для серіалізації, десеріалізації, загальний час та розмір файлу (див. рис. 4.6). У таблиці 4.4 записані середні значення для третього випробування.

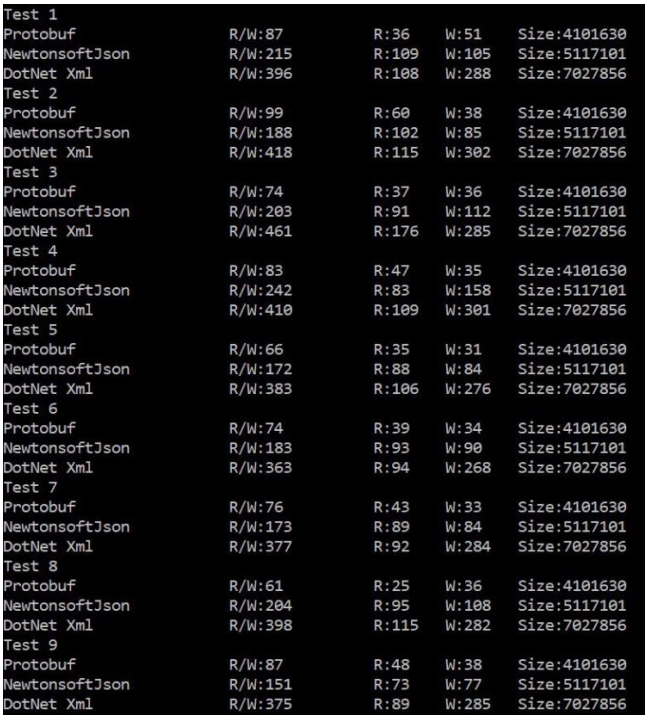

Рис. 4.6 Результати третього випробування

Таблиця 4.4

Усереднені результати третього випробування

Метод	Час серіалізації, мс	Час десеріалізації, мс	Загальний час, мс	Розмір файлу, байт
Protobuf	36,9	41,1	78,6	4101630
JSON	100,3	91,4	192,3	5117101
XML	285,7	111,6	397,9	7027856

Як видно з узагальнених результатів третього випробування, різниця за часом серіалізації, десеріалізації та загальним часом між JSON та XML майже в 2 рази. В той же час найкращі результати отримано використовуючи Protobuf.

Четверте випробування

У другому випробувані вхідні дані складаються з 50000 рядків. Після кожної ітерації, як у попередніх випробуваннях, фіксуємо час, який необхідний для серіалізації, десеріалізації, загальний час та розмір файлу (див. рис. 4.7). У таблиці 4.5 записані середні значення для другого випробування.

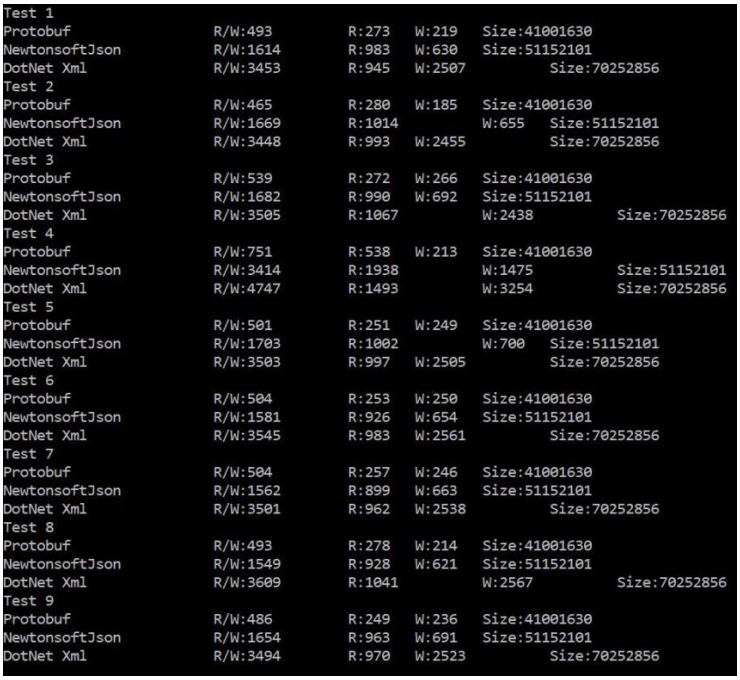

Рис. 4.7 Результати четвертого випробування

Таблиця 4.5

Усереднені результати четвертого випробування

Метод	Час серіалізації, мс	Час десеріалізації, мс	Загальний час, мс	Розмір файлу, байт
Protobuf	230,9	294,6	526,2	41001630
JSON	753,4	1071,4	1825,3	51152101

Продовження таблиці 4.5

Метод	Час серіалізації, мс	Час десеріалізації, мс	Загальний час, мс	Розмір файлу, байт
XML	2594,2	1050,1	3645	70252856

Аналізуючи результати четвертого випробування видно, що найкращі результати за всіма критеріями отримано використовуючи Protobuf. Як видно з таблиці 4.5, в цьому випробуванні XML має найбільший час серіалізації, десеріалізації, загальний час та розмір файлу. В порівнянні з JSON, час серіалізації Protobuf менше в 3,27 разів, час десеріалізації менше в 3,64 разів, загальний час менше в 3,5 разів, а розмір файлу менший в 1,25 разів.

4.2. Оцінка запропонованого методу серіалізації

Для проведення порівняльної оцінки побудуємо графіки за визначеними критеріями.

- час, необхідний для серіалізації вхідних даних

За результатами чотирьох випробувань у таблиці 4.6 показані узагальнені результати часу серіалізації для трьох методів. На рисунку 4.8 представлений графік часу серіалізації в залежності від кількості рядків вхідного файлу

Таблиця 4.6

Час серіалізації вхідних даних

Кількість рядків вхідного файлу	Protobuf (мс)	JSON (мс)	XML (мс)
50	25,4	37,1	25,6
500	16,8	24,7	33,2
5000	36,9	100,3	285,7
50000	230,9	753,4	2594,2

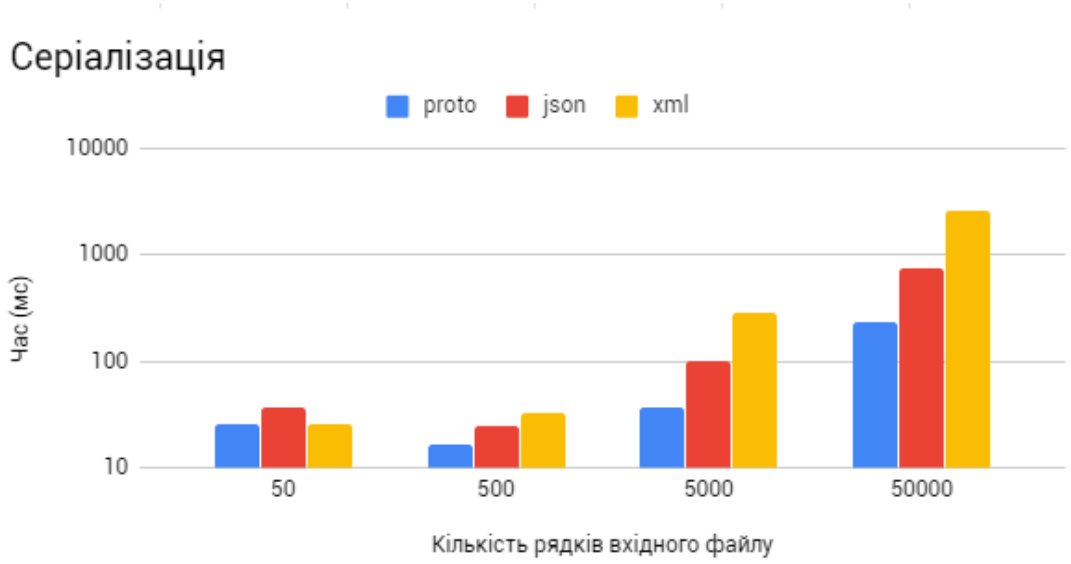

Рис. 4.8 Час серіалізації в залежності від кількості рядків вхідного файлу

Як видно з графіку, у різних випробування час серіалізації, використовуючи Protobuf, найменший в порівнянні з JSON та XML. Час серіалізації XML більший за JSON, окрім першого випробування.

- час, необхідний для десеріалізації серіалізованих даних

За результатами чотирьох випробувань у таблиці 4.7 показані узагальнені результати часу десеріалізації для трьох методів. На рисунку 4.9 представлений графік часу десеріалізації в залежності від кількості рядків вхідного файлу

Таблиця 4.7

Час десеріалізації вхідних даних

Кількість рядків вхідного файлу	Protobuf (мс)	JSON (мс)	XML (мс)
50	0,7	6,4	2,7
500	4,1	11,9	14,2
5000	41,1	91,4	111,6
50000	294,6	1071,4	1050,1

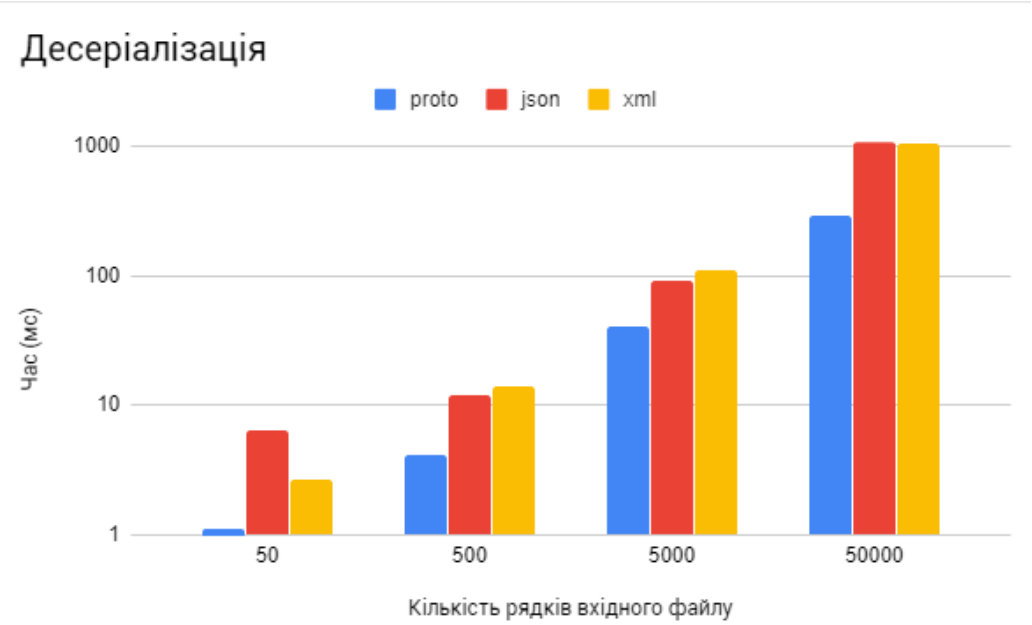

Рис. 4.9 Час десеріалізації в залежності від кількості рядків вхідного файлу

Як видно з графіку, у чотирьох випробування час десеріалізації, використовуючи Protobuf, найменший в порівнянні з JSON та XML. В першому випробуванні час десеріалізації XML значно менший за JSON, у решті випадків він майже однаковий.

- час, необхідний для серіалізації та десеріалізації вхідних даних

За результатами чотирьох випробувань у таблиці 4.8 показані узагальнені результати часу серіалізації та десеріалізації для трьох методів. На рисунку 4.10 представлений графік часу серіалізації та десеріалізації в залежності від кількості рядків вхідного файлу.

Таблиця 4.8

Час серіалізації та десеріалізації вхідних даних

Кількість рядків вхідного файлу	Protobuf (мс)	JSON (мс)	XML (мс)
50	27,1	44,3	28,5
500	21,3	37	48
5000	78,6	192,3	397,9
50000	526,2	1825,3	3645

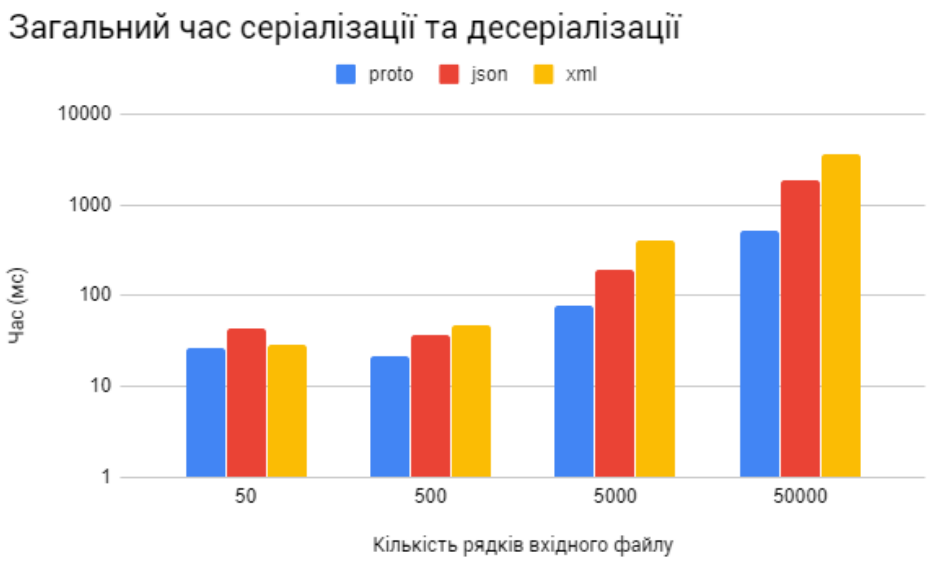

Рис. 4.10 Час серіалізації та десеріалізації в залежності від кількості рядків вхідного файлу

Як видно з графіку, у різних випробування загальний час серіалізації та десеріалізації, використовуючи Protobuf, найменший в порівнянні з JSON та XML. Проте в першому випробуванні Protobuf майже не відрізняється від XML, хоча у інших випробуваннях, зі збільшенням кількості рядків вхідних даних, різниця збільшується.

- розмір серіалізованого файлу

За результатами чотирьох випробувань у таблиці 4.9 показані узагальнені результати розмірів серіалізованих файлів для трьох методів. На рисунку 4.11 представлений графік розмірів серіалізованих файлів в залежності від кількості рядків вхідного файлу.

Таблиця 4.9

Розміри серіалізованих файлів

Кількість рядків вхідного файлу	Protobuf (кбіт)	JSON (кбіт)	XML (кбіт)
50	41,6	52	71,4
500	402	501,6	688,8
5000	4005,5	4997,2	6862,1
50000	40040,7	49953,2	68606,3

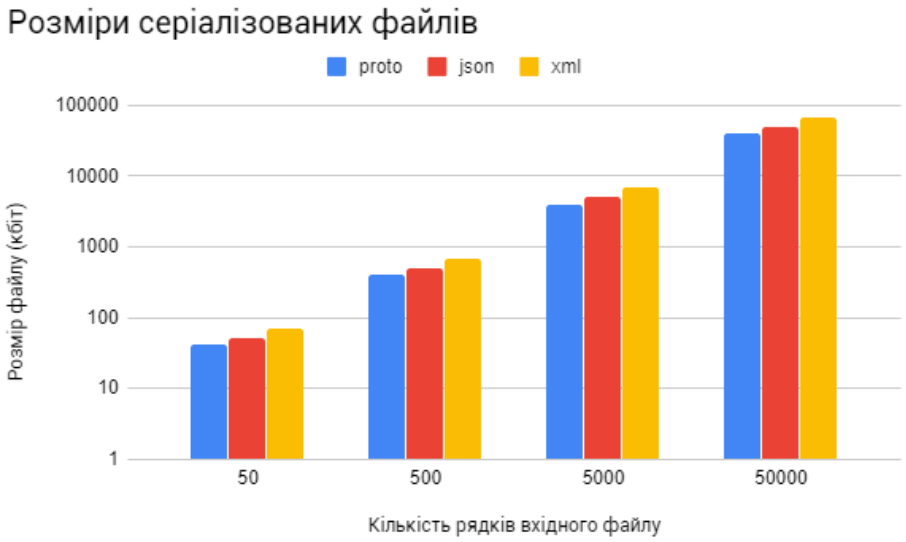

Рис. 4.11 Розмір серіалізованих файлів в залежності від кількості рядків вхідного файлу

Як видно з графіку, у всіх чотирьох випробування розмір серіалізованого файлу, використовуючи Protobuf, найменший в порівнянні з JSON та XML. З іншого боку, використовуючи XML, отримано найбільші розміри серіалізованого файлу у всіх випробуваннях в порівнянні з Protobuf та JSON.

Математичний опис отриманих результатів

В залежності від обраного критерію для методу серіалізації Protobuf можна сформувати математичні вирази на основі отриманих у кожному випробуванні узагальнених результатів. Використовуючи спосіб знаходження проміжних значень за відомим дискретним набором значень [27], отримаємо формули:

1. Для часу серіалізації, десеріалізації та загального часу

$$t = b + \left(\frac{n-n_0}{n_1-n_0}\right) * (a-b), \quad n_0 \leq n < n_1 \qquad (4.1)$$

де t – час серіалізації / десеріалізації / загальний час, n - кількість рядків вхідного файлу, n_0 – початкова кількість рядків вхідного файлу, n_1 – максимальна кількість рядків вхідного файлу, a – час серіалізації / десеріалізації / загальний час при максимальній кількості рядків вхідного

файлу, b – час серіалізації / десеріалізації / загальний час при початковій кількості рядків вхідного файлу.

2. Для розміру серіалізованого файлу

$$s = b + \left(\frac{n-n_0}{n_1 - n_0}\right) * (a-b), \quad n_0 \le n < n_1 \tag{4.2}$$

де s – розмір серіалізованого файлу, n - кількість рядків вхідного файлу, n_0 – початкова кількість рядків вхідного файлу, n_1 – максимальна кількість рядків вхідного файлу, a – розмір серіалізованого файлу при максимальній кількості рядків вхідного файлу, b – розмір серіалізованого файлу при початковій кількості рядків вхідного файлу.

Отримані математичні вирази дозволять швидко розраховувати час серіалізації, десеріалізації та загальний час, а також розмір серіалізованого файлу в залежності від кількості вхідних даних.

За результатами моделювання отримані наступні математичні вирази:

1) Час серіалізації

$$t = \begin{cases} 0{,}508*n, & 0 \le n < 50 \\ 25{,}4 - 1{,}91*10^{-2}*(n-50), & 50 \le n < 500 \\ 16{,}8 + 4{,}467*10^{-3}*(n-500), & 500 \le n < 5000 \\ 36{,}9 + 4{,}311*10^{-3}*(n-5000), & 5000 \le n < 50000 \end{cases} \tag{4.3}$$

2) Час десеріалізації

$$t = \begin{cases} 0{,}014*n, & 0 \le n < 50 \\ 0{,}7 + 7{,}556*10^{-3}*(n-50), & 50 \le n < 500 \\ 4{,}1 + 8{,}222*10^{-3}*(n-500), & 500 \le n < 5000 \\ 41{,}1 + 5{,}633*10^{-3}*(n-5000), & 5000 \le n < 50000 \end{cases} \tag{4.4}$$

3) Загальний час серіалізації та десеріалізації

$$t = \begin{cases} 0{,}542*n, & 0 \le n < 50 \\ 27{,}1 - 1{,}289*10^{-2}*(n-50), & 50 \le n < 500 \\ 21{,}3 + 1{,}273*10^{-2}*(n-500), & 500 \le n < 5000 \\ 78{,}6 + 9{,}947*10^{-3}*(n-5000), & 5000 \le n < 50000 \end{cases} \tag{4.5}$$

4) Розмір серіалізованого файлу

$$s = \begin{cases} 0{,}832*n, & 0 \le n < 50 \\ 41{,}6+0{,}8*(n-50), & 50 \le n < 500 \\ 402+0{,}801*(n-500), & 500 \le n < 5000 \\ 4005{,}5+0{,}801*(n-5000), & 5000 \le n < 50000 \end{cases} \quad (4.6)$$

Висновки:

1) В четвертому розділі проведено моделювання запропонованого методу серіалізації за обраними критеріями, яке підтвердило ефективність та працездатність запропонованого рішення.
2) Проведено оцінку запропонованого методу серіалізації та сформовані математичні вирази на основі отриманих у кожному випробуванні узагальнених результатів, які дозволяють визначити час та розмір серіалізованого файлу для заданої кількості рядків вхідного файлу.

ЗАГАЛЬНІ ВИСНОВКИ ПО РОБОТІ

1) Розглянуто основні відмінності IIoT і IoT, що дало змогу сформувати особливості побудови IIoT та виявити необхідність удосконалення процесу збору та обробки інформації.
2) Проведено аналіз проблем IIoT та проаналізовано існуючі способи збору та обробки інформації в мережі IIoT на основі чого обрано прототип методу збору та обробки інформації.
3) Модифіковано метод збору та обробки інформації, який відрізняється застосуванням серіалізації на основі методу Protobuf, що дозволяє підвищити ефективність обробки інформації за часовими показниками та зменшити об’єм переданої інформації.
4) Проведено натурне моделювання запропонованого рішення за рахунок реалізації програмного забезпечення на основі модифікованого методу. Особливістю є фіксація параметрів - часу та розміру серіалізованого файлу. Дане моделювання підтвердило працездатність запропонованого рішення.
5) За результатами моделювання проведено оцінку запропонованого рішення - в порівнянні з JSON, час серіалізації Protobuf менше в 3,27 разів, час десеріалізації менше в 3,64 разів, загальний час менше в 3,5 разів, а розмір файлу менший в 1,25 разів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Internet of Things in the 5G Era: Enablers, Architecture, and Business Models [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/7397856.
2. Internet of Things: Applications and Challenges in Technology and Standardization [Електронний ресурс]. – 2011. – Режим доступу до ресурсу: https://www.researchgate.net/publication/51890865_Internet_of_Things_Applications_and_Challenges_in_Technology_and_Standardization.
3. Internet of Things in Industries: A Survey [Електронний ресурс] // IEEE Transactions on Industrial Informatics 10(4):2233-2243. – 2014. – Режим доступу до ресурсу: https://www.researchgate.net/publication/270742269_Internet_of_Things_in_Industries_A_Survey.
4. The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0 [Електронний ресурс] // IEEE Industrial Electronics Magazine. – 2017. – Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/7883994.
5. The Industrial Internet of Things Volume G1: Reference Architecture [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://www.iiconsortium.org/IIC_PUB_G1_V1.80_2017-01-31.pdf.
6. Industrial IoT Dataflow and Security Architecture [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: http://cdn.ttgtmedia.com/rms/IoTAgenda/PracticalIndustrialInternetofThingsSecurity-Chapter2.pdf.
7. Three Industrial IoT Implementation Challenges [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://dzone.com/articles/3-iiot-industrial-internet-of-things-implementatio.
8. How to Solve Common Challenges of IIoT in Manufacturing [Електронний ресурс]. – 2017. – Режим доступу до ресурсу:

https://www.einfochips.com/blog/how-to-solve-common-challenges-of-industrial-iot/.

9. Big Data in IoT [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.researchgate.net/publication/338365769_Big_Data_in_IoT.
10. A reliable Internet of Things based architecture for oil and gas industry [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/7890184.
11. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks [Електронний ресурс] – Режим доступу до ресурсу: https://tools.ietf.org/html/rfc6550.
12. Development of an industrial Internet of things suite for smart factory towards re-industrialization [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://link.springer.com/article/10.1007/s40436-017-0197-2.
13. How Fog Computing Can Support Latency/Reliability-sensitive IoT Applications [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://books.google.com.ua/books?redir_esc=y&hl=ru&id=xf7XuQEACAAJ&q=144#v=snippet&q=144&f=false.
14. A middleware based architecture for the industrial internet of things [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://www.researchgate.net/publication/306160243_A_middleware_based_architecture_for_the_industrial_internet_of_things.
15. Wireless sensor networks: a survey on recent developments and potential synergies [Електронний ресурс]. – 2013. – Режим доступу до ресурсу: https://link.springer.com/article/10.1007/s11227-013-1021-9.
16. ZigBee, Bluetooth and Wi-Fi Complex Wireless Networks Performance Increasing [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://www.researchgate.net/publication/316746554_ZigBee_Bluetooth_and_Wi-Fi_Complex_Wireless_Networks_Performance_Increasing.
17. Smart home products with Z-Wave [Електронний ресурс] – Режим доступу до ресурсу: https://www.z-wave.com/learn.

18. Исследование протоколов взаимодействия Интернета вещей на базе лабораторного стенда [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://www.sut.ru/doci/nauka/review/20161/55-67.pdf.
19. MQTT in IoT [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.verypossible.com/blog/what-is-mqtt-in-iot.
20. Защита IoT-данных на уровне сети/транспорта [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://www.ibm.com/developerworks/ru/library/iot-trs-secure-iot-solutions2/index.html.
21. Serialization [Електронний ресурс] – Режим доступу до ресурсу: https://www.techopedia.com/definition/867/serialization-net.
22. Protocol Buffers [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com/protocol-buffers/docs/overview.
23. Protobuf Performance Comparison [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://dzone.com/articles/protobuf-performance-comparison-and-points-to-make.
24. IoT Development Needs Microservices and Containerization [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://www.einfochips.com/blog/why-iot-development-needs-microservices-and-containerization.
25. Аналіз методів серіалізації об’єктів для побудови платформи великих індустріальних даних / Д. Попенко, В. Курдеча // Перспективи телекомунікацій/ Д. Попенко, В. Курдеча. – Київ, 2020. – С. 222–224.
26. Застосування Protobuf в індустріальному Інтернеті речей / Демид Попенко // Перспективи розвитку інформаційно-телекомунікаційних технологій та систем / Демид Попенко. – Київ, 2020. – С. 369.
27. Интерполяция [Електронний ресурс]. – 2005. – Режим доступу до ресурсу: https://upload.wikimedia.org/wikipedia/commons/4/43/Kazakhstan_National_encyclopedia_%28ru%29_-_Vol_2_of_5_%282005%29.pdf.